

PRACTICAL INVESTIGATIONS IN ROBOT LOCALIZATION USING ULTRA-  
WIDEBAND SENSORS

BY

NICHOLAS ALBERT WYATT WRIGHT

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Mechanical Engineering  
in the Graduate College of the  
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Adviser:

Professor Geir E. Dullerud

## Abstract

Robot navigation is rudimentary compared to the capabilities of humans and animals to move about their environments. One of the core processes of navigation is localization, the problem of answering where one is at the present time. Robot localization is the science of using various sensors to inform a robot of where it is within its environment. Ultra-wideband (UWB) radio is one such sensor technology that can return absolute position information. The algorithm to accomplish this is known as multilateration, which uses a collection of distance measurements between multiple robot tag and environment anchor pairs to calculate the tag's position. UWB is especially suited to the task of returning precise distance measurements due to its capabilities of short duration, high amplitude pulse generation and detection. Decawave Ltd. has created an UWB integrated circuit to perform ranging and a suite of products to support this technology. Claimed and verified accuracies using this implementation are on the order of 10cm. This thesis describes various experiments carried out using Decawave technology for robot localization. The progression of the chapters starts with commercial product verification before moving into development and testing in various environments of an open-source driver package for the Robot Operating System (ROS), then the development of a novel phase difference of arrival (PDoA) sensor for three-dimensional robot localization without an UWB anchor mesh, before concluding with future research directions and commercialization potential of UWB. This thesis is designed as a compilation of all that the author has learned through primary and secondary research over the past three years of investigation. The primary contributions are:

1. A modular ROS UWB driver framework and series of ROS bags for offline experimentation with multilateration algorithms.
2. A robust ROS framework for comparing motion capture system (MoCap) ground truth vs sensor data for rigorous statistical analysis and characterization of multiple sensors.
3. Development of a novel UWB PDoA sensor array and data model to allow 3D localization of a target from a single point without the deployment of an antenna mesh.



## Dedication

*To all the amazing individuals with whom I have crossed paths with at the University of Illinois Urbana-Champaign. We came from the world over to learn, live, and love in the American Midwest. Few of us will remain, but the bonds we built together are here to stay.*

## Acknowledgements

Thank you to the U.S. Army Corps of Engineers, Engineering Research and Development Center, Construction Engineering Research Lab for supporting this research. Thank you to Professor Geir Dullerud for advising me throughout my research. Thank you to Dan Block for teaching me about mechatronics and controls. Thanks to Clinical Associate Professor Bob Norris for supporting research on the Wiser system. Special thanks to Decawave for providing PDoA hardware and taking the time to answer my questions.

## Table of Contents

CHAPTER 1: Introduction .....	1
CHAPTER 2: Testing the Decawave DW1000 Enabled WISER Locator System.....	20
CHAPTER 3: Developing a ROS-Based Experimental Infrastructure .....	34
CHAPTER 4: Small-Scale Decawave TREK1000 UWB Testing.....	51
CHAPTER 5: Large-Scale Decawave TREK1000 UWB Testing.....	59
CHAPTER 6: Trimble SX-10 Total Station Localization .....	71
CHAPTER 7: Single-Anchor UWB Localization using Multiple Differential PDoA Antenna Pairs..	81
CHAPTER 8: Future Research .....	121
CHAPTER 9: UWB Commercialization Potential .....	124
CHAPTER 10: Conclusion.....	127
REFERENCES .....	128
APPENDIX A: GitHub Repository .....	138

## CHAPTER 1: Introduction

*Where am I?* Determining one's location and orientation in space is one of humankind's oldest problems. Inextricably linked to survival, the need to know where home is and where food is located define a dependence on position in the environment. Necessary to answer the question are maps to reference from. Humans have been creating their own maps since prehistoric times: "Some of the cave paintings and other representations on bones and artifacts, which used to be viewed as mere artistic representations, have turned to be, according to the latest investigations, maps of hunting areas, streams, routes, and even maps of the stars" [1]. The study of maps is known as cartography and it is a principally human endeavor. "As mediators between an inner mental world and an outer physical world, maps are fundamental tools helping the human mind make sense of its universe at various scales...and the mapping experience...undoubtedly existed long before the physical artifacts we now call maps" [2].

### Astronomy

Aside from early manmade maps, there are the environmental landmarks that have been used for wayfinding throughout human evolution, such as memorable rocks, terrain, trees, and other landscape markers. Looking beyond the terrestrial, stars are a map that have been present from the start of humanity. Astronomy is among the original natural sciences and the first to reach a high level of mathematical and predictive sophistication [3]. Babylonian astronomers by the end of the second and beginning of the first millennium BCE had identified the existence of planets as distinct from the other stars and objects in the night sky [4]. The astronomical tradition and field continued advancing over the next 4000 years as a leading science until the present day [3]. Star charts, also known as star maps, have been a part of astronomy since time immemorial. The first recorded star chart may be at least 32,500 years old as a carving of the constellation Orion [5]. This is all to say that astronomy is one of humankind's oldest endeavors and ties together with the history of localization and mapmaking [6].

### Navigation

Besides survival, maps and localization were fundamental to exploration of new territories. Civilization has been exploring the seas using celestial navigation since at least 3000 BCE when

the Minoans sailed around Crete [7]. Since then, various navigational aids have been developed to assist in exploring first the oceans, then air, and now space. Some of the most notable instruments have been the magnetic compass to measure the Earth's field, the astrolabe and sextant to measure latitude, the chronometer to measure time, and modern tools such as the gyroscopic compass, radar, and GNSS, which will be discussed shortly [8].

Humans have invented countless tools to use in tandem with maps to determine time and place for navigation, however these have been innovations rather than innate. Indeed, outside of the normal range of human perception, there are numerous methods of navigation available to other living organisms in the world that science has found and has yet still to uncover. Some of these include magnetic field detection, celestial navigation using the sun, moon, and stars, and skylight polarization tracking [9]. Methods less connected to the Earth and stars include scent finding [10] and general landmark recognition using cognitive maps [11].

## Robot Localization

Unlike living organisms, robots have not had the convenience of billions of years of evolution to refine their survival and navigation instincts [12]. Instead, they must rely on the ingenuity of engineers to design sensors, actuators, and algorithms to define their place in the universe. Although robot navigation involves multiple sub-fields of robotics including perception, localization, cognition, and motion control, this thesis focuses on the question of robot localization [13]. Analogous to the human endeavor, "robot localization provides an answer to the question: *Where is the robot now?*" [14] To answer this question, a robot must be provided with sensor measurements to determine its current position and orientation in the environment. Thus, "robot localization is the problem of determining the pose of a robot relative to a given map of the environment" [15].

Any rigid body has at most six degrees of freedom (DoF) that fully and uniquely define its position and orientation in space. The standard ROS geometry message, "Pose", combines a cartesian linear point position with a quaternion angular orientation about the point and this thesis will use pose in the same definition [16].

All sensors provide either absolute or relative measurements. Absolute sensors provide measurement data directly referenced to an environment that the robot is navigating about. Relative sensors, on the other hand, provide measurement data referenced from a coordinate frame that is attached to the robot itself [17]. A special type of absolute sensor provides measurement data referenced from a global coordinate frame. This global coordinate frame is typically attached to the Earth, however it can also be any frame which can be deterministically calculated in relation to Earth, such as another planet. This definition contrasts with the calculation of the transformation between a map frame and earth frame because the map frame may be attached to a ship, plane, or other environment which can move independently from Earth.

ROS REP 105 provides definitions for the parent/child relationships between its standard frames of earth, map, odometry (odom), and base\_link [18]. For the purposes of the current discussion, absolute sensors will return measurements referenced from the map frame and relative sensors return measurements referenced from the odom frame. The relationship between the map and odom frames is that odom's pose is continuous and drifts over time relative to map. This drift is caused by dead reckoning, the act of determining current pose from a prior pose by the integration of relative sensor measurements taken since that prior pose [19]. Dead reckoning can provide accurate pose estimates for a short amount of time, depending on the accuracy of the relative sensors, but will always drift from true pose over time due to the integration of sensor errors [20].

The problem of a drifting odom frame is mitigated by combining absolute sensor information with the dead reckoned pose to come up with a more accurate estimate of robot pose. There will always be some amount of error in the estimated robot pose because all sensors introduce measurement noise, however state noise relative amplitude varies based on sensor quality. There are several methods of estimating robot pose from a fusion of multiple sensor measurements, including extended and unscented Kalman filtering as well as particle filtering [14], however these methods are not the focus of our discussion. Let us now take a closer look at some of the sensors available for robot localization.

## Sensors for Robot Localization

This thesis is about UWB, but to understand why this technology is relevant one must first know some of the other options available, along with their strengths and limitations. This is a brief introduction to some of the most pertinent and interesting sensors used for robot localization.

### Wheel Odometry [21]

Wheel odometry utilizes rotary encoders to measure the motion of a robot's wheels or some other rotary mechanism that is converted into linear motion of the robot. Using the radii of the wheels or another conversion constant of the robot along with the number of rotary encoder elements, motion of the robot can be calculated. This is a classic example of dead-reckoning, whereby the robot tries to count how many "steps" it has taken since its last known position. The method works best for linear motion, however there is always a degree of error present due to the requirement of wheel slip necessary for motion, wheel wear, surface conditions, etc. For differential drive robots, those with more than two wheels especially, executing turns necessitates excessive wheel slippage and further degrades wheel odometer measurements. For these reasons, robot localization carried out solely via wheel odometry will experience unbounded positional error growth. Wheel odometry does have the benefit of fast sampling times, which makes it a standard sensor for robot localization.

### Inertial Measurement Units [21]

Inertial measurement units (IMUs) are a system of sensors designed to provide full six DoF information for determining the current robot pose relative to a pose in the past. Typical systems include at least triaxial accelerometers to measure linear accelerations and triaxial rate gyroscopes to measure angular velocities. Other sensors may also be present, such as triaxial magnetometers to absolutely determine orientation relative to Earth's magnetic field, altimeters to measure vertical displacement from sea level, and thermometers to compensate for any temperature-dependent sensor variations. IMUs may even include an internal Kalman filter to provide enhanced pose messages. Although absolute orientation is theoretically possible to measure using magnetometers, robots and environments usually have high levels of magnetic interference relative to the strength of Earth's magnetic field. Due to the double integration required to calculate position from acceleration and the single integration to

calculate orientation from angular velocity, IMU dead reckoning also provides unbounded pose estimates over time relative to a known starting position. Sampling rates can be as fast as those required for motor control loops and this allows updated pose information in real-time. Along with wheel odometry, IMUs are typically the baseline system for performing robot localization.

### Ultrasonic

Ultrasonic sensors utilize a sound transducer to send and receive high-frequency (ultrasonic) pressure waves. These acoustic waves travel through air and reflect to the sensor unit when they encounter a surface. The sensor compares the time of transmittal to the time of receipt of the signal and uses the speed of sound to calculate the distance traveled. This allows object detection in a cone expanding from the sensor unit. Various sensing distances are available from at least 2cm to 5m range [22]. Ultrasonic sensors are convenient for coarse obstacle detection and can detect some clear materials that infrared-based sensors cannot, but there are fundamental limitations that limit their usefulness for robot localization. Some of these include lower fidelity at longer distances, on/off state that cannot detect obstacle sizes, tendency to lose reflected signal when obstacles are at an angle to redirect the acoustics, and reflection characteristics dependent on surface finish [23]. As air is the ultrasonic medium, wind can introduce external disturbances for this sensor type [24]. Overall, ultrasonic sensors are crude obstacle detectors, but poor object identifiers, which are necessary for accurate robot localization. They have fallen out of use in recent years, in favor of more accurate sensors.

### Radar

Like ultrasonic sensors, radar has historically been used as a coarse obstacle detection sensor for robots. One of the most popular radar sensors of the past decade has been Delphi's electronically scanning radar (ESR) that was released in 2009 and set the standard for automotive long-range radar [25]. Unlike traditional mechanically rotating radar, this unit utilizes what Delphi calls Simultaneous Transmit and Receive Pulse Doppler (STAR PD) to perform forward obstacle detection using a solid-state unit [26]. It combines a long-range mode for detection up to 174m within a cone of 20 degrees and mid-range mode for detection up to 60m away within a cone of 90 degrees. Although the Delphi ESR can detect multiple discrete obstacles' position, velocity, and acceleration, it returns only estimated centroid values for each



without any size or shape information. This precludes its use as a mapping sensor and other automotive radars are the same story. Many uses of automotive radar, such as blind spot detection, require only an on/off signal as to whether an obstacle has been detected or not. There have been efforts to improve radar sensor information for robot localization to perform map-based feature detection rather than just obstacle detection. Most of these studies use solid state radar [27], but there are many studies using rotating radar as well [24], [28], [29]. As a result of advances in signal processing using machine learning and similar algorithms, radars for robot localization now have better knowledge of object dimension, orientation, and classification that can be used for map-based localization but should still be paired with other sensor modalities for accurate map-based localization [30].

### Map-Based Localization

Rather than a single sensor type, map-based localization is a set of algorithms that utilize a priori maps of the environment or Simultaneous Localization and Mapping (SLAM) for robot localization [21]. The basic form this type of localization uses occupancy grid maps to discretize the environment into cells that are either occupied with obstacles or deemed safe to navigate through. More sophisticated versions identify features within the map and use a process called loop-closure to refine the robot's pose based on proximity to these map features. "Loop closing is the act of correctly asserting that a vehicle has returned to a previously visited location" [31]. This technique allows the robot to adjust its absolute pose based on where it has been before. This class of localization algorithms variously use Kalman filters, Monte Carlo methods, Markov processes, and particle filters to probabilistically estimate the robot's pose in relation to the mapped environment and its prior poses [32].

Sensors that typically enable map-based localization are cameras and 3D light detection and ranging (lidar) [21]. Lidar systems use either a solid-state or rotating array of laser beams that can measure distances calculated from reflected signal strength. Lidar densities have increased enough to allow machine learning based object recognition and feature extraction able to semantically categorize obstacles in the map rather than just marking them as obstacles. Lidar has good range and resolution, but costs are generally high, and performance suffers greatly in adverse weather. Various types of cameras may also be utilized as SLAM sensors, either on

their own or paired with lidar. Monocular or stereo cameras are possible, in black & white or full RGB color spectrum. Thermal or infrared cameras are also another possible sensor modality with strengths in adverse weather and contrast compared to visual spectrum cameras. No matter the type of camera used, image classification techniques are used to identify and recognize features within the environment to map and perform loop-closure upon. [30].

### Sky Tracking

Robot localization has historically neglected the heavens that humans have relied on to navigate. Objects in the sky can provide global absolute pose measurements, given an accurate clock and a priori information on sky object location over time. The objects to be discussed here are the sun and stars. These are especially useful sensors in an extra-terrestrial application due to the lack of GNSS systems and detailed terrain maps outside of our planet.

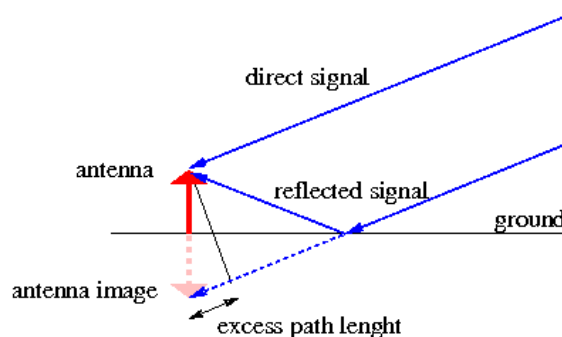
During the daytime, the sun is available for viewing by an appropriate sensor. Even if it is cloudy out, it is possible to use a polarization sensor to determine the location of the sun in the sky [33]. The method used for bearing determination is to sense where the sun is in a sensor array, look up in an astronomical almanac where the sun is supposed to be located at the present time, and calculate the robot's heading compared to true North or another reference [34]. It is important to correct the robot frame using the gravity vector from an IMU to approximate a tangent plane at the current location on the surface of the planet [35]. The sun sensor can be a light detector using a lens of sufficient convexity to see the entire sky or a camera-based sextant [36].

At night, the stars are visible with sufficient intensity to be detected by electronic sensors. The method of detection is the same as with the sun, however now a pattern matching system is needed to determine which stars are being detected. The algorithm proceeds similarly as the solar sensor, but now there is more than one object that can be detected. Thus, triangulation is possible by using multiple altitude and azimuth angles and comparing to a local star chart at the present date and time [37]. This can enable absolute pose estimation without any other sensors required, although wheel encoders and IMU are sure to be in use as well.

## Global Navigation Systems

Global navigation satellite system (GNSS) is a generic name for constellation satellite navigation systems that provide geo-spatial positioning with global coverage. Specific instances of GNSS are the Global Positioning System (GPS), GLONASS, Galileo, and Beidou [38]. These systems provide global absolute position information with accuracies ranging from millimeters to meters [21]. Many robotic systems throughout the world rely on GNSS, but it has several drawbacks. The update rate to get a position fix is orders of magnitude slower than that of relative sensors such as wheel odometry and IMUs. The solution to this discrepancy in refresh rate is to employ sensor fusion, such as a Kalman filter to give continuous pose information that can be updated when discrete position data arrives. This method also solves the issue of sensor drift with relative sensors only, because GNSS provides an absolute position.

GNSS relies on direct line of sight (LOS) between the satellite constellation and the receiver. This fact precludes accurate or sometimes any operation in applications and areas with sky blockage, such as indoors, under heavy tree canopy, caves, and urban environments with tall buildings. Complete blockages of the sky will prevent GNSS from returning measurements and partial blockages will result in signal deterioration from reflections and multipath fading. See Figure 1.1 for an explanation of how signal reflections degrade accuracy. Further complicating the use of GNSS sensors for robot localization are the existence of relatively inexpensive jammers and spoofers [39], which preclude the use of GNSS as the sole localization sensor in military robotics or any application with adversaries.



*Figure 1.1: Reflected signals appear to have traveled a further distance than direct signals. [40] The antenna picks up the reflected signal and has no way of knowing whether it is receiving a direct signal or reflected signal. If both are received, then it can choose the shortest distance and/or strongest signal. If only a reflected signal is received, the distance will be measured to be greater than actuality and accuracy will suffer as a result.*

Although GNSS has become ubiquitous over the past twenty years, it was not the first global navigation system nor is it the only still in existence. As a result of the relative ease of jamming and spoofing GNSS, the US government has released a mandate to provide an alternative by 2023 [41]. One of the potential solutions is Enhanced Loran (eLoran), which itself is an update to the since decommissioned Loran-C system. These are land-based radio systems that follow in the tradition of OMEGA and DECCA, two previous land-based radio navigation systems used for submarine, ship, and airplane navigation. These types of systems all utilize high-power, low-frequency radio transmissions from known tower locations to perform time difference of arrival (TDoA) localization. Their availability is based on where towers are placed, so global coverage would necessitate radio towers throughout the world [42]. eLoran is an alternative position, navigation, and timing (PNT) to GNSS and complements it. With accuracies around 10 meters, the system is not accurate enough alone for robot localization, but it is ground-based and can penetrate buildings and water with resilience to jamming and spoofing. While GNSS signals are low-power and high-frequency, eLoran is the high-power and low-frequency, which makes the signal millions of times stronger than GNSS [43]. Figure 1.2 nicely sums up these distinctions in a graphic. These ground-based global localization systems are included both for historical reasons and as an additional sensor to be aware of in the future.



Figure 1.2: “eLoran is a high power, Low Frequency (LF), ground wave radio broadcast system, capable of providing 10-20 meter positioning accuracy, Stratum-1 frequency distribution, and Universal Time Coordinated (UTC) timing well within one microsecond ( $\mu$ s) across very large areas (1,000 miles)” [44].

## Ultra-Wideband (UWB) Radio

Ultra-wideband is so named due to wide spectrum of radio that it occupies compared to conventional narrowband and broadband radio signals [45]. See Figure 1.3 for a conceptual illustration of the relative spectrum widths and amplitudes of these three signal categories.

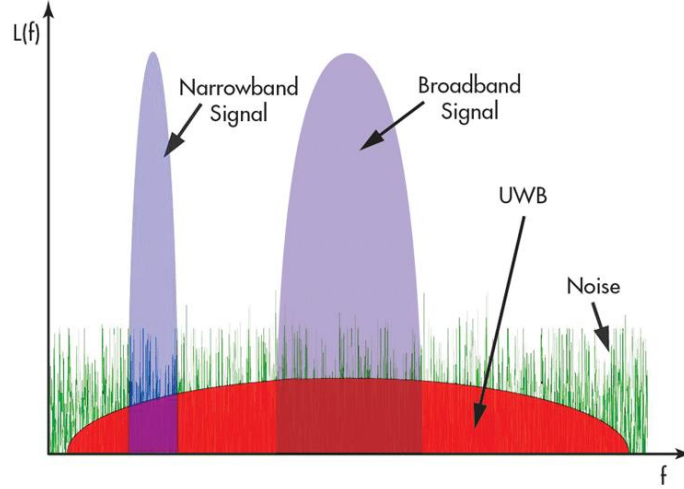


Figure 1.3: A visual comparison of frequency bandwidths and signal strengths. Ultra-wideband occupies such a wide range of frequencies that it must coexist with existing radio transmission standards, but under the amplitude that is defined as the noise floor for those existing signals. [46]

According to the Federal Communications Commission (FCC) revision of its Part 15 limits that legalized UWB, the definition of UWB is a bandwidth defined between two -10dB emission points of greater than or equal to 500 MHz or a fractional bandwidth of larger than 0.2 [47]. The -10 dB cutoff emission points are known as  $f_H$  for the higher frequency and  $f_L$  for the lower, see Figure 1.4. Absolute bandwidth,  $B$ , is calculated as in equation (1.1).

$$B = f_H - f_L \quad (1.1)$$

The UWB center frequency,  $f_c$ , is calculated as in equation (1.2).

$$f_c = \frac{f_H + f_L}{2} \quad (1.2)$$

The fractional bandwidth,  $B_{frac}$ , is finally defined as in equation (1.3).

$$B_{frac} = \frac{2(f_H - f_L)}{f_H + f_L} = \frac{B}{f_c} \quad (1.3)$$

See Figure 1.4 for a visual diagram of these definitions within a frequency spectrum.

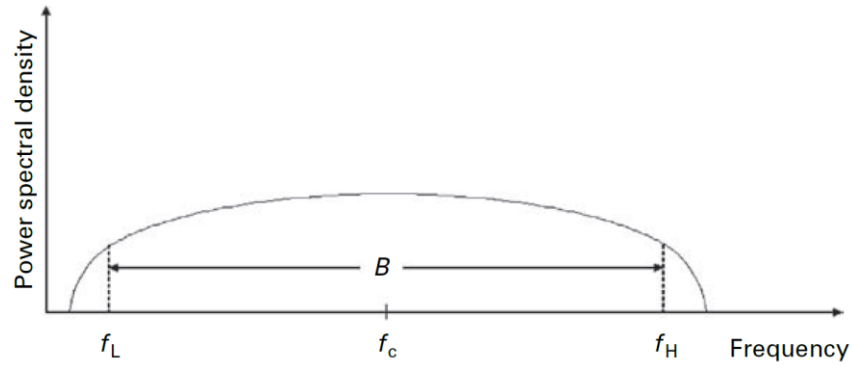


Figure 1.4: A UWB signal is defined to have an absolute bandwidth  $B \geq 500\text{MHz}$ , or a fractional bandwidth  $B_{\text{frac}} > 0.2$ . [45]

Figure 1.5 shows the U.S. UWB spectrum as defined by the FCC in relation to some other commonly used signals. Most notable in the diagram is the overlap between the 802.11a spectrum (5 GHz Wi-Fi) and the UWB spectrum. Spectrum overlap can occur because the FCC has allowed UWB only as an unlicensed emission below the Part 15 noise limit that exists for low-power, non-licensed transmitters such as various consumer electronics devices [48].

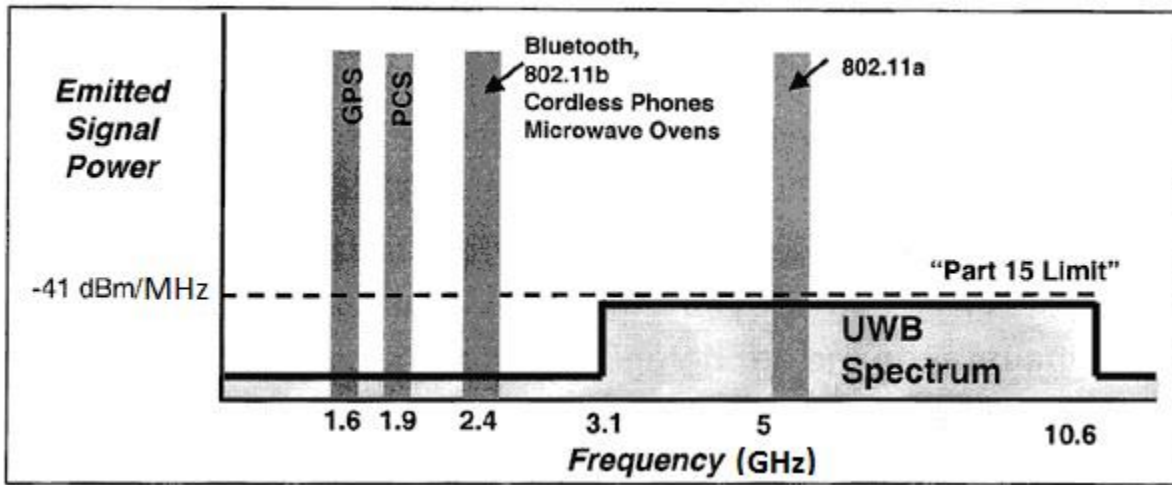


Figure 1.5: Comparison between US narrowband and UWB radio spectrum. 802.11b is the common 2.4 GHz Wi-Fi frequency and 802.11a is the 5 GHz Wi-Fi frequency. The Personal Communications Service (PCS) frequency is commonly used for mobile voice and data services in cell phones. As indicated, UWB signal power exists below the part 15 noise floor that all unlicensed transmitters must fall below. [49]

As a result of their large bandwidth, UWB systems utilize short duration pulses for communication. This fact results from the Fourier transform pair of the  $\text{rect}()$  and  $\text{sinc}()$  functions, see Figure 1.6. Frequency domain bandwidth is inversely proportional to time domain period of the waveform. Also, for a given frequency domain signal amplitude, the time domain pulse amplitude is directly proportional to the frequency domain bandwidth. This

property is what allows the entire UWB spectrum to exist below the Part 15 noise floor, but still generate detectable signals in the time domain. UWB pulse durations are typically on the order of one nanosecond [45]. Practically, UWB waveforms are not made of  $\text{sinc}()$  functions, but the Fourier transform analogy remains instructive. Actual UWB pulse signals can be made using Gaussian [50], Hermitian, Laplacian, Rayleigh [51], or wavelet pulses [45].

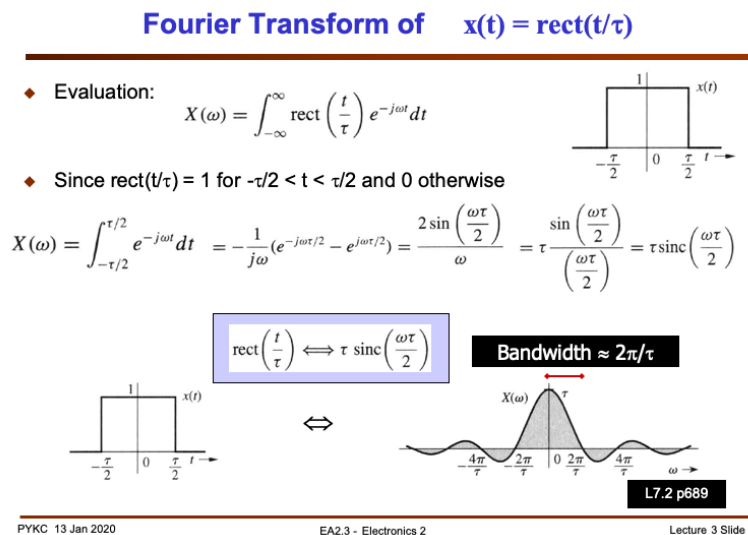


Figure 1.6: The Fourier transform pair between a  $\text{rect}()$  and a  $\text{sinc}()$  function. Pulse duration is inversely proportional frequency bandwidth. Pulse amplitude is directly proportional to bandwidth, given a fixed bandwidth amplitude. [52]

Extremely short pulse durations make UWB an ideal candidate for distance ranging applications. Narrowband technologies, when used for localization, all present the same fundamental deficiency regarding reflections and multipath propagation. See Figure 1.7 for a visualization of the phenomena where signals overlap in time relative to their period duration. Despite practical implementations of Wi-Fi, Bluetooth, Zigbee, and other narrowband technologies for indoor localization, none have accuracies better than on the order of 1m due to this physical limitation [53]. Even GNSS experiences the same practical issues, as a narrowband technology, but accurate operation is generally understood to be restricted to clear skies. As discussed earlier, GNSS localization accuracy degrades in cluttered environments. This is the same case with all narrowband technologies, which operate optimally in uncluttered environments, but real-world indoor applications are typically far from this ideal.

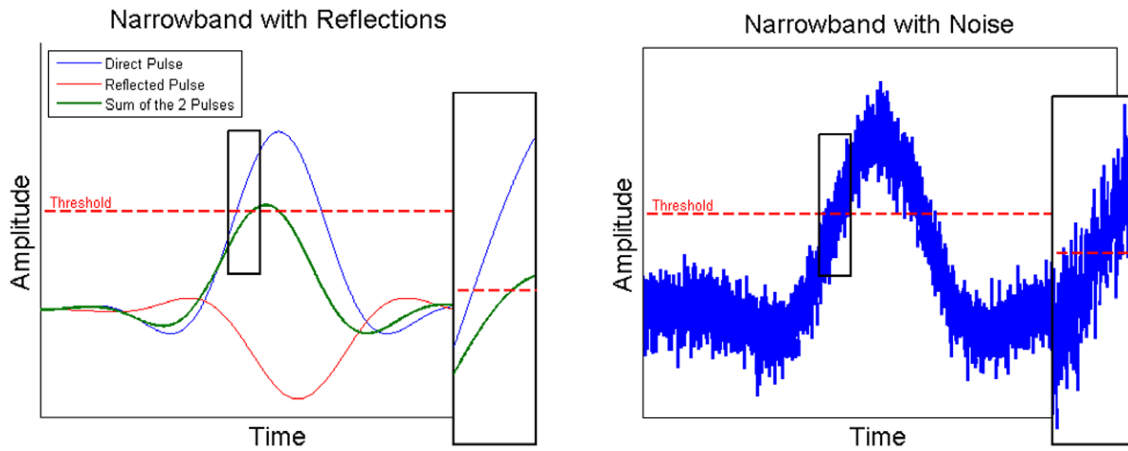


Figure 1.7: Narrowband systems are vulnerable to multipath interference and result in positioning accuracy of  $\geq 1\text{m}$  in real-world implementations. Narrowband signals are low bandwidth in the frequency domain and of long duration in the time domain. Even when reflected signals have lower energy, they still combine into one signal read at the receiver with a peak amplitude not far from that expected from the primary signal alone. This makes it difficult to discern between first path returns and multipath reflections when there is real-world noise present in the measurements. [54]

UWB excels in cluttered environments that result in signal reflections. See Figure 1.8 for a visualization of how this works. Short time duration pulses occupy short physical distances when traveling through medium and multipath reflections can be efficiently separated as discrete signals by the receiver. This characteristic is what theoretically makes UWB an improved radio technology for localization compared to narrowband solutions. In practice, UWB does perform well, delivering localization accuracies on the order of 10cm, an order of magnitude improvement over narrowband systems.

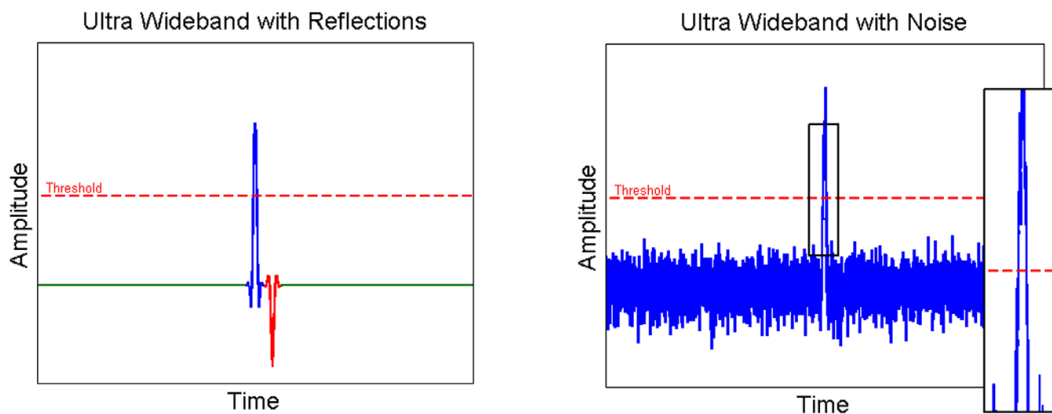


Figure 1.8: UWB systems are resilient to multipath interference and thus enable positioning accuracy of  $\geq 10\text{cm}$  in real-world conditions. Short-duration pulses can be separated and identified at the receiver because primary and secondary signal amplitude peaks are expected to be relatively far from each other in time. Multipath signal interference is thus not a prevalent phenomenon to confuse the receiver. [54]



There are problems with UWB that have delayed its ubiquity. Notably, there are the antenna design difficulties that come along with supporting such a wide spectrum of frequency transmission and receive. Antennas are often ignored or assumed ideal in narrowband system analysis, but this is not a valid assumption for UWB systems. Narrowband antennas are designed to be resonant at a single center frequency [55]. In contrast, UWB antennas act as a signal filter that differentiates both at the transmitter and receiver. This is one of the reasons that Gaussian pulses are used as UWB signals, because their first derivative is a monocycle and the second derivative is a doublet, as can be seen in Figure 1.9. Some of the issues that arise with UWB antennas are linearity, radiation efficiency, and impedance matching across the entire frequency band [55]. There are now many geometries of practical UWB antennas available as the area of research boomed after the FCC regulated UWB in 2002 [56].

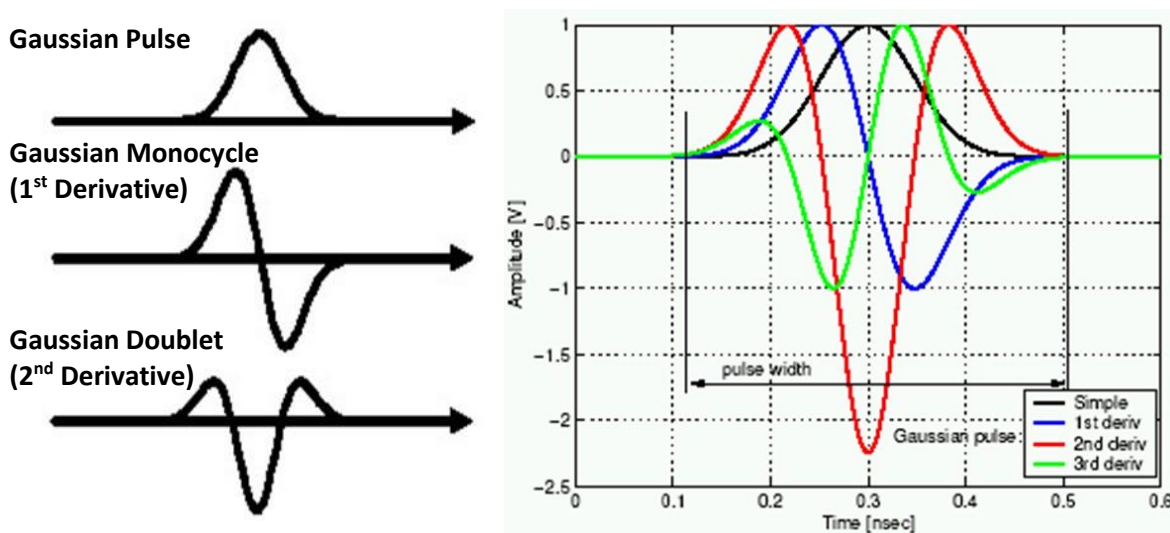


Figure 1.9: Visual depiction of a Gaussian pulse and its two first order derivatives. This signal is attractive for use in UWB systems due to its analytical tractability, feasibility of generation, and desirable characteristics of derivatives compared to the parent signal. [57]

UWB is a valuable technology for robot localization. Whereas GNSS is the de facto standard for outdoor localization, UWB may soon be the standard for indoor navigation. Compared to some of the other sensors that have been discussed, UWB is one that requires infrastructure installation in the environment rather than on the robot. This situation is analogous to that of GNSS, which now only requires a receiver to operate on a device, but first needed satellites launched. The cost of setting an UWB system up is initially high to calibrate its position but it can operate in the background after that. This makes the technology ideal for static built environments such

as airports, shopping malls, grocery stores, stadiums, convention centers, parking garages, and densely packed urban metros. These are areas that humans as well as robots would benefit from an absolute localization system. Indeed, Decawave was recently acquired by Qorvo, a supplier to Apple, who themselves added UWB to the iPhone 11 prior to the acquisition [58]. Enabling Google Maps inside just as outside could be revolutionary from a human navigation perspective. UWB is not just for indoors, however. The technology works just as well outside and can provide absolute localization in unique environments such as underground caves or mines, densely canopied forests, extra-terrestrial planets, and other GNSS-denied locales. The system setup may be more complicated in these cases than attaching UWB anchors to walls, but it is also possible to install them on dynamic pieces of the environment, such as other robots, drones, people, and equipment. If the anchors are in a known position, the receiving tag can perform multilateration, as discussed in the next section. Finally, UWB can augment other robot localization sensors, as it has unique characteristics as a radio-based technology. Fog, rain, dust, snow, and other environmental factors create noisy signals for vision and light-based sensors [59]. Radio is more immune to these factors and UWB can still operate in such conditions.

The Shannon-Hartley theorem, equation (1.4), says that channel capacity,  $C$ , measured as a data rate, is linearly proportional to the signal bandwidth and logarithmically proportional to the mean-square sign-to-noise ratio [60].

$$C = B \log_2 \left( 1 + \frac{S}{N} \right) \quad (1.4)$$

This property makes UWB desirable for high-capacity data communications and in fact the technology is a major component of emerging next-generation cellular phone technology and is variously called 5G Plus, 5G UWB, or Ultra Capacity 5G by each of the three large U.S. carriers AT&T, Verizon, and T-Mobile, respectively [61]. All these names are synonymous for millimeter-wave (mmWave) technology [62], which is UWB at higher radio frequencies than are being discussed for localization in this thesis. Although Marconi's spark gap radios were sending UWB signals in 1901 [45], it has taken another 100 years to develop practical data transmittal systems due to the relative difficulty of encoding information using pulse signals and the

challenge of designing hardware to do so [63]. The reader is left to pursue more detailed sources about the history [64] and technical aspects [65] of UWB on their own.

### Time of Flight Localization

The process of determining location from a series of distance measurements is known as multilateration. “Multilateration is a localization technique based on measuring the difference of the distances between the robot and landmarks” [66]. The literature often refers to multilateration as trilateration, which is defined as “the measurement of the lengths of the three sides of a series of touching or overlapping triangles on the earth's surface for the determination of the relative position of points by geometrical means” [67]. Furthermore, the process of trilateration is often conflated with triangulation, which is also used for determining unknown positions, but triangulation specifically refers to the process of using angles rather than the distances that trilateration uses. The reader is likely to most encounter triangulation used in everyday speech and trilateration referenced in literature. These definitions are used inconsistently throughout the literature. Both multilateration and trilateration will be used in this thesis to denote the process of determining a position from multiple distance measurements.

Distance-based localization algorithms do not typically measure distance directly, but rather use the propagation time of an electromagnetic wave in medium and the corresponding speed of light,  $c$ , to calculate distance. This method is known as time of flight (ToF) and utilizes the instantaneous measures of time at two antennas to determine the time interval that has elapsed between transmission and receipt of a signal between the two [42, p. 2]. Two ToF-based algorithms are time of arrival (ToA) and time difference of arrival (TDoA). Each method utilizes geometric relationships created based on the ToF distances measured [42, p. 169]. Both algorithms can be found implemented in UWB systems and each has its practical strengths and weaknesses. The current discussion will focus on ToA as the ToF method used in this thesis.

The ToA algorithm uses distances directly in to calculate an unknown position. These distances can come from any source method, but ToF is generally more accurate than others. ToA utilizes the geometry of circles in 2D, or spheres in 3D, to create a system of equations for determining

the location of an unknown point relative to multiple anchors in the map with known locations. Figure 1.10 shows the geometry of the situation in 2D.

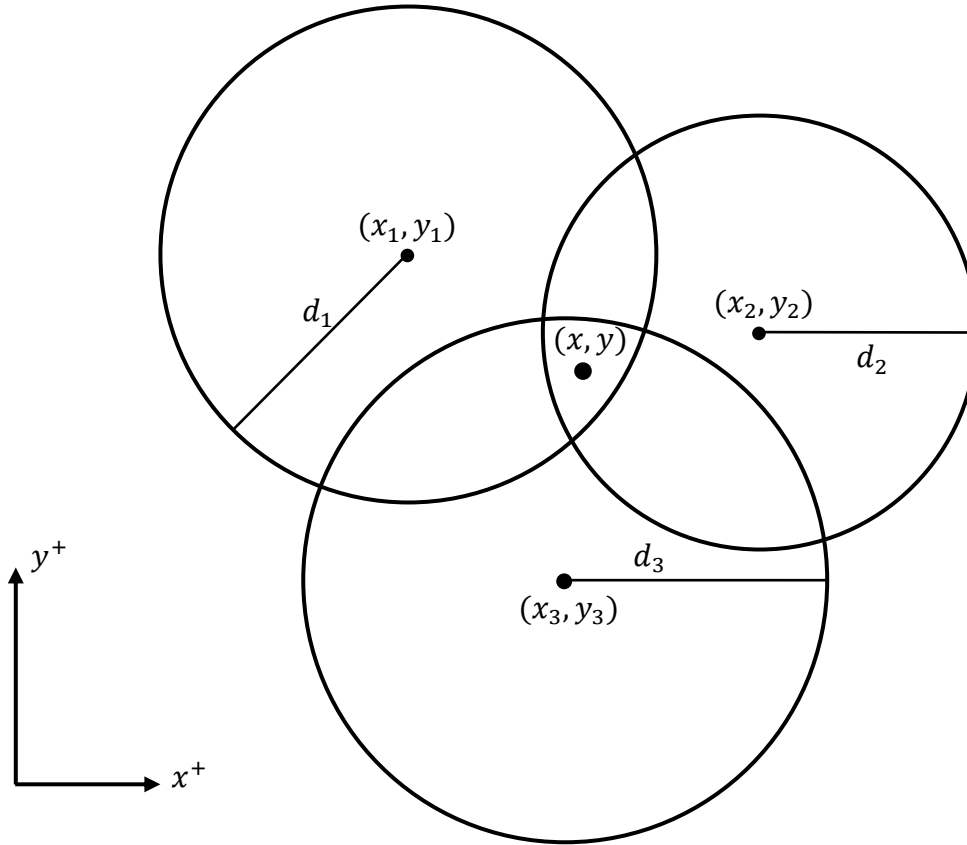


Figure 1.10: A diagram of the 2D trilateration problem with noisy measurements. The known ToA distances measured from the anchors numbered 1-3 are radii of circles that indicate the possible locus of locations for the unknown point  $(x, y)$ . Because the ToA measurements include noise, the three circles do not intersect exactly at the point of interest.

$$\begin{aligned}
 d_1^2 &= (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 \\
 d_2^2 &= (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 \\
 d_3^2 &= (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 \\
 d_4^2 &= (x - x_4)^2 + (y - y_4)^2 + (z - z_4)^2 \\
 &\vdots \\
 d_n^2 &= (x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2
 \end{aligned} \tag{1.5}$$

System of equations (1.5) describes the  $n$  possible equations used to solve the ToA localization problem. Due to quadratic nature of the equations, there will be two solutions if using the minimum number of anchors required for localization, thus for an unambiguous solution, 2D localization requires at least three anchors (thus the name trilateration) and 3D localization

requires at least four anchors. However, if altitude is known, three anchors are typically all that are required for localizing in 3D space. For example, in GNSS positioning, the second solution is far above the surface of the Earth and can be discarded, as shown in Figure 1.11 and Figure 1.12. A similar case often occurs in indoor positioning systems when using ground robotics that remain in the plane of the lower solution. As can also be seen in Figure 1.10, each ToA measurement comes with noise that precludes a perfect geometric solution to the problem. For this reason, some type of non-linear optimization or iterative solution method is required to solve equation (1.5).

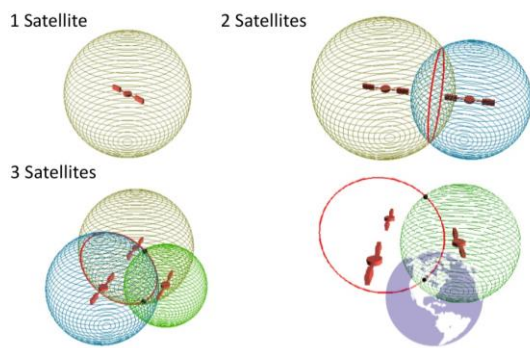


Figure 1.11: Geometric schematic of theoretical GPS positioning [68]. Using only three satellites yields two solutions, one of which is far above the surface of the Earth and can be safely discarded.

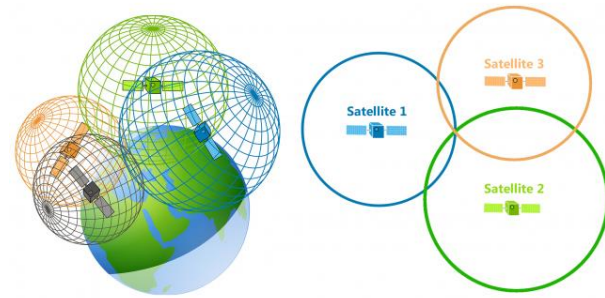


Figure 1.12: Planar view of the intersection point of three distances from GPS positioning satellites [69]. The addition of another ToA measurement from a fourth satellite geometrically constrains the localization solution.

ToA localization using ToF requires calculating the time interval between the sender and receiver, but this requires either clock synchronization between the anchors or two-way communication between the tag and anchors. The GPS positioning system achieves anchor clock synchronization by using accurately tuned atomic clocks in an open loop configuration and an additional ToA measurement to solve for the clock offset at the receiver. Any other clock-based system will require the same procedure or two-way communication to obtain accurate ToA measurements without the requirement of an extra ToA equation. Two-way communication eats up system bandwidth and limits the number of tags able to interact with a set of anchors in each time interval. An alternative is the TDoA algorithm that can support an unlimited number of tags. TDoA uses time differences of arrival and leads to a hyperbolic set of equations to solve for localization. The details of this algorithm are left for the reader as none of the research here utilized this method. The Crazyflie Loco positioning system is one practical

example of an implemented UWB TDoA system [70]. Joao Porto's TREK1000 driver code also implements TDoA and is recognized here as a foundation for creating the TREK1000 ToA driver code found in this thesis [71].

The reader is directed to Alan Bensky's consistent and readable text [42] to learn further theoretical and implementation details about the algorithms discussed here and many more.

### Phase of Arrival Techniques

There are multiple methods and meanings evoked when referring to phase of arrival (PoA) [53], [42], [72]. These methods can be used to calculate distance or angle of arrival (AoA), depending on the implementation. The similarity between the methods is that they employ techniques to measure the phase of an arriving signal and compare it to the phase of a reference signal to calculate a phase difference of arrival (PDoA). Figure 1.13 shows a relationship between the distance of two antenna arrays and a signal's arriving angle. This measurement can be used to calculate a phase PDoA or a TDoA, depending on the needs of the system. PoA can also be combined with other methods to improve localization. We will discuss this method further in CHAPTER 7: Single-Anchor UWB Localization using Multiple Differential PDoA Antenna Pairs by looking at Decawave's specific implementation of PDoA to calculate 2D position in a plane.

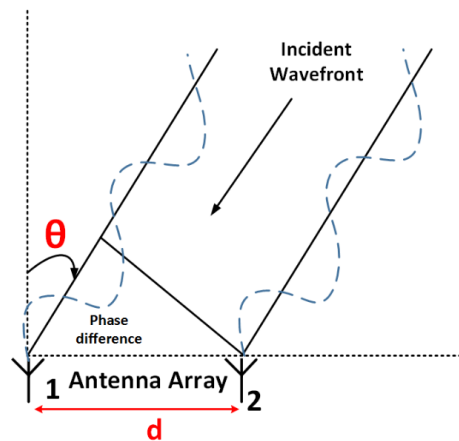


Figure 1.13: Schematic representation of phase difference of arrival (PDoA) measurement. The distance between two antennas is related to the arrival angle of a signal by the PDoA of the signal at each antenna.

## CHAPTER 2: Testing the Decawave DW1000 Enabled WISER Locator System

This chapter is a preliminary study on UWB, conducted at the start of the research effort in summer 2018, that provided the experimental foundation for the remainder of the thesis. Although Decawave UWB accuracy claims were verified, not much more could be done due to the standalone nature of the system. The following several chapters refine and expand on the work initiated here.

### Executive Summary

The goal of this study was to determine whether the WISER Locator system is appropriate for use in robot localization. To make this determination, an OptiTrack optical positioning system was used to obtain ground truth location data in the millimeter level accuracy range, an order of magnitude more accurate than current UWB systems can achieve. A small, autonomous car was directed around a circuit with the OptiTrack system collecting position data multiple times per second. Simultaneously, the WISER Locator system was collecting position data at the same location on the vehicle. The collected data was analyzed for accuracy and precision by comparing the Wiser Locator to the OptiTrack system.

### Results

Results collected verify the published literature on the accuracy of the Decawave DW1000 UWB localization chip [73], [74], [75], [76]. Standard positional accuracy in this study was determined to be  $\pm 10\text{cm}$  for the WISER Locator system. This is an order of magnitude more accurate than differential GPS [77] and is highly suited to be used in conjunction with an inertial measurement unit (IMU) and magnetometer in determining robot pose. Although the WISER system was sufficient for use in a research lab setting, it currently has shortcomings that preclude its use in autonomous robotics applications:

#### 1. wiser.exe

The application must be running on Windows to obtain data from the system. WISER Locator should include support for command line control in Ubuntu along with a ROS driver to support native use in the ROS ecosystem.

## 2. Static antenna requirement

The mesh network must be setup with one of the static antennas connected to the computer running wiser.exe. The best solution would be to offload the data processing into embedded processors on the antenna modules and then have dynamic antennas communicate position data directly to the robot.

## 3. Physical realization

The current antenna housings are designed for application on walls. Their awkward shape makes mounting challenging in any other situation. The housings should be re-designed as waterproofed, flanged boxes with the transceiver location clearly marked and documented.

## Recommendation

The results of this study agree with the published literature on the viability of using UWB based localization systems in robotics applications [78], [79], [80]. Until WISER addresses the listed shortcomings, implementing the Locator system will require too much unique setup within the robotics application to be generally feasible. Rather, two other recommended implementations of the Decawave DW1000 chip UWB technology are the evaluation modules from Decawave themselves [81] or the Pozyx system [82]. Both are currently less expensive than WISER's system and are designed to be used in robotics applications. Furthermore, the Pozyx system has a proven track record in peer-reviewed research [83], [84], [85].

## Procedure

The WISER system was initialized using five static antennas: four located close to  $\frac{3}{4}$  of the distance up each of the four walls in the UIUC Mechatronics Laboratory, along with the fifth antenna positioned closer to the volumetric center of the lab. The four wall mounted antennas had their support bases taped in place and then antennas mounted and connected via an extension cable into USB power adapters plugged into wall outlets below each location. The fifth more centrally located antenna was connected to the host Windows PC and acted as the central data aggregator for all the other static and dynamic antennas.



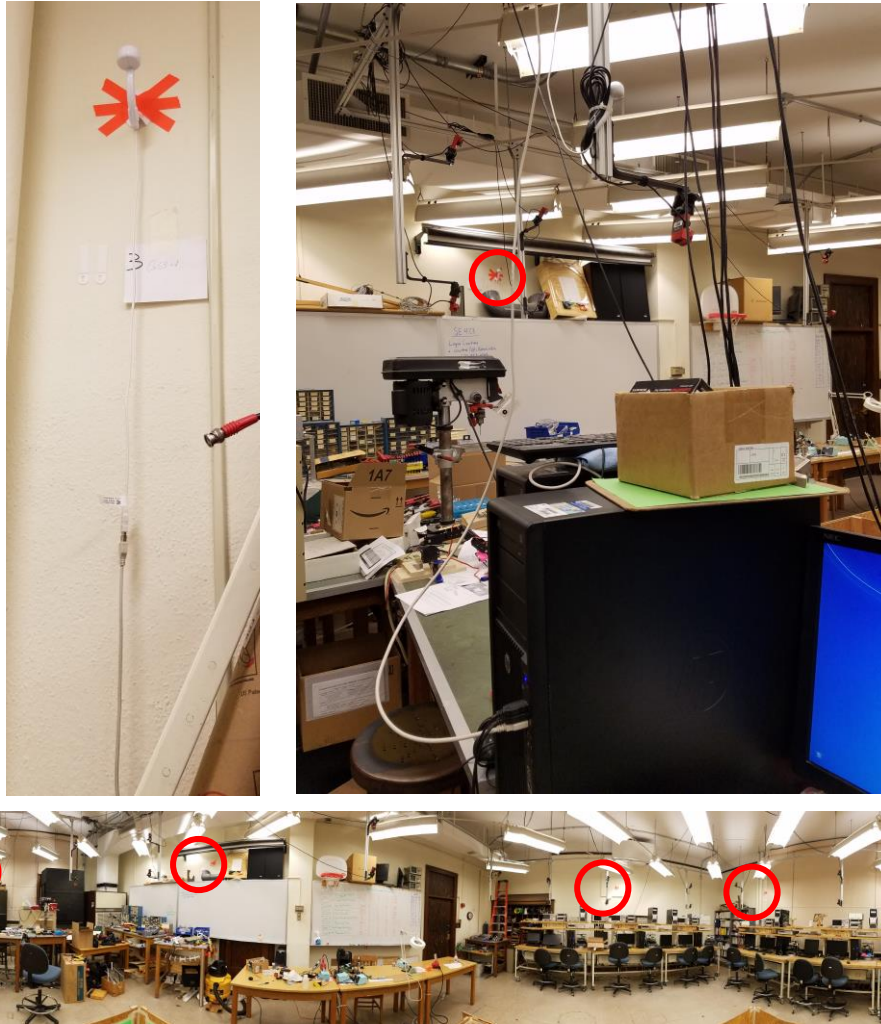


Figure 2.1: Physical connections and locations of the 5 static antennas making up the WISER Locator mesh network. Upper left shows a typical wall mounted antenna with USB cord hanging and connected to an extension USB plugged into a wall outlet. Upper right shows the mounting for the static antenna connected to the host PC. One wall mounted antenna is circled in the background. Lower frame shows all 5 static antennas circled in a panorama of the laboratory.

The dimensions of the room were assumed to be rectangular and measured by three independent methods:

1. Counting the number of 12" square tiles from wall to wall in each direction.
2. Using a tape measure (which was shorter than either dimension of the room).
3. Using a laser measurement device.

Each of the methods had its difficulties. The tiles were not all whole. The baseboard interfered with getting the tape measure from wall-to-wall horizontally. The laser was difficult to hold perpendicular to the reference surface. All methods bore similar results, but it was decided that laser measurement was the most accurate and precise method. All locations of the static

antennas were then measured from a reference wall and the floor to the center of the bottom flat part of the antenna. All these measurements were recorded in the room layout picture at a scale of ½ inch/pixel.

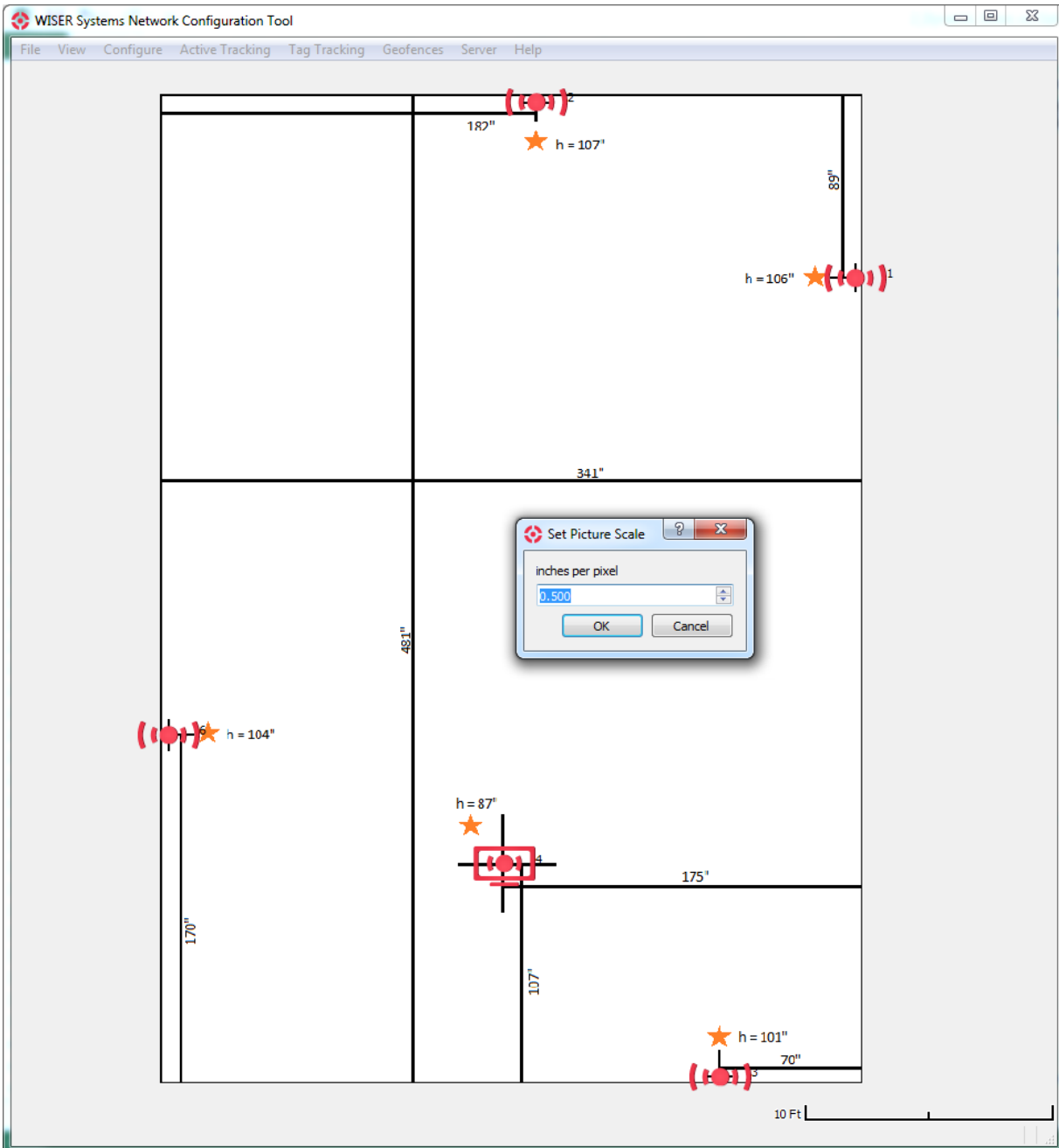
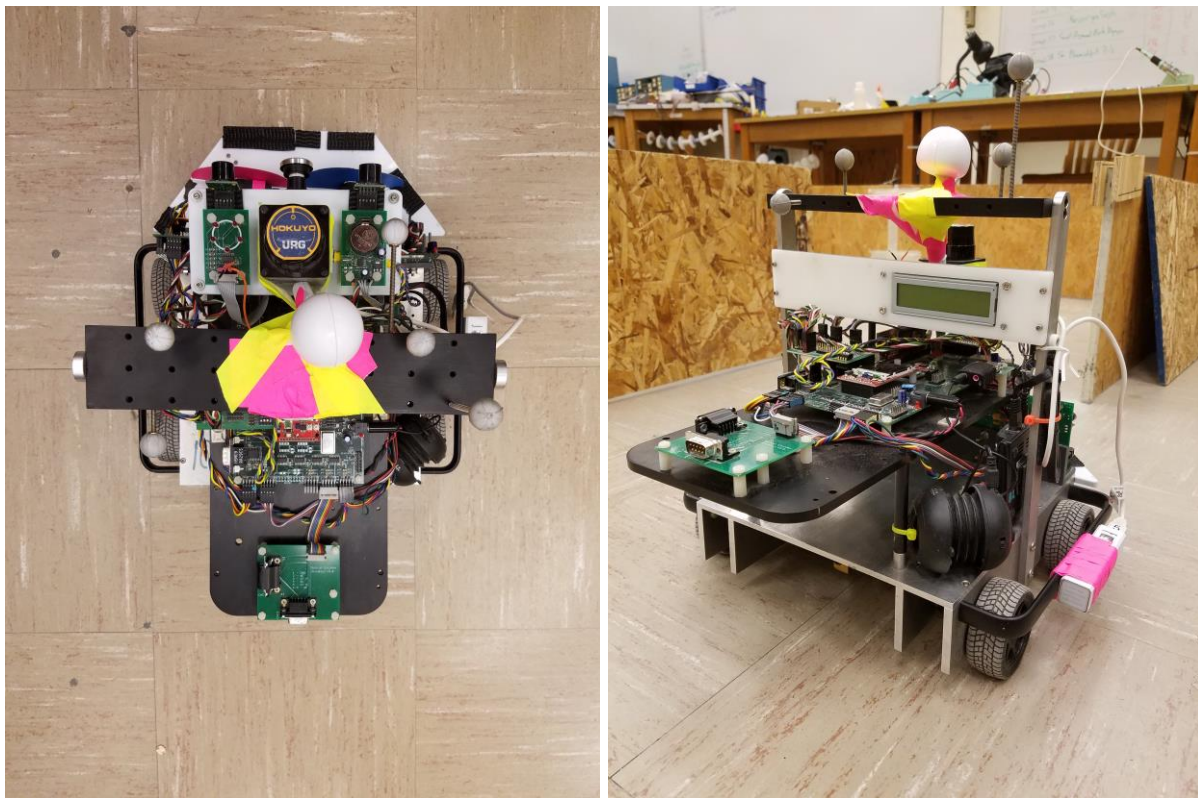


Figure 2.2: Physical setup of the wiser antennas translated into a picture to be used in wiser.exe. The room was scaled into a picture at the rate of 0.5 inches per pixel and all relevant measured dimensions of the setup were hardcoded into a .png file. Wiser.exe asks for the layout picture and scale and then updates its settings .xml file based on the location that each antenna is dragged to in the GUI. The stars do not mean anything other than an antenna is located at the nearby red dot.

The WISER mesh setup was initialized using the directions provided with the system and referring to the independent study report by Adithya Jaikumar [86]. The room layout picture was loaded and each of the antennas was set to calibration for identification and then dragged to its correct location based on the crosshairs in the layout picture created from measurements.

Subsequent to setup of the static WISER mesh network, a sixth active antenna node was powered on through connection to a USB battery pack. This antenna was initialized as an active tracking tag and left affixed at the centroid in the XY plane created by the OptiTrack retroreflectors on one of the Mechatronics robotic cars. The selected antenna position allowed XY data to be directly compared between the two systems without the need for a rigid body coordinate frame transformation.



*Figure 2.3: Antenna attached to the robotic car as an active tag to be tracked by the WISER system. Left frame shows the placement of the antenna within the cluster of OptiTrack retroreflectors. Right frame further illustrates antenna mounting to the robotic car. Lithium-ion battery pack can be seen attached to the car's wheel guard and connected to the antenna.*

There are a few other settings that must be correctly adjusted to obtain good data. These are highlighted in the settings caption below.

```
<?xml version="1.0"?>
<Wiser>
  <Map file="C:/Users/dan5/Desktop/NickW_Wiser/x64/calibration/New_Mechatronics_0.5_inches_per_pixel.png" scale="0.5" />
  <Server tls="false" />
  <Settings tdoafilter="1" tag_height="12" />
  <Network>
    <Anchor shortid="4" longid="0x10205f6b10001468" x="166.957" y="106.333" z="87">
      <Anchor shortid="1" longid="0x10205f6b100017e3" x="337.779" y="391.319" z="106" />
      <Anchor shortid="2" longid="0x10205f6b10001449" x="182.486" y="476.849" z="107" />
      <Anchor shortid="3" longid="0x10205f6b10001813" x="271.751" y="3.30806" z="101" />
      <Anchor shortid="6" longid="0x10205f6b10001478" x="4.31269" y="169.425" z="104" />
    </Anchor>
  </Network>
</Wiser>
```

Figure 2.4: Final wiser.exe settings used in obtaining position data from the system. Antenna heights can be specified with the GUI, but XY locations are from a drag-and-drop procedure which introduces error into their measurements. A better procedure is to directly edit the settings .xml file when setting XY locations. The setting "tdoafilter" specifies the number of points to filter using a moving average. This should be set to 1 in the settings .xml file to turn off filtering as the application does not always respond to menu toggles and settings. Tag height refers to the fixed distance of the active tag from the common ground plane. This was set statically for the robot car scenario, but this should not be set for a changing height tag.

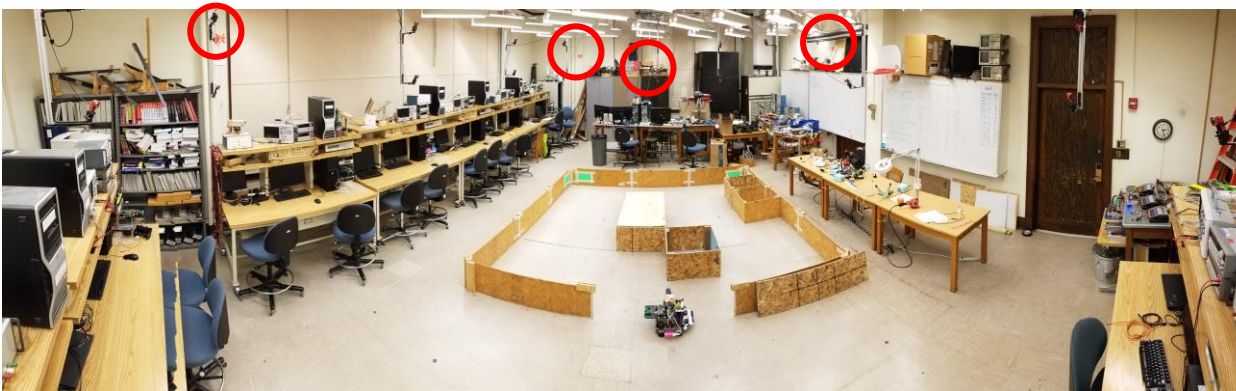


Figure 2.5: Robotic car at the entrance to the course and facing along the positive X-axis. Positive Y is toward the far wall. The course can be seen in reference to the rest of the Mechatronics laboratory. The visible WISER antennas are circled.

Once the mesh network was configured and active tracking turned on for the car antenna, the WISER system took on a passive data collection role while the robot car drove in a circuit around the course. The mechatronics car already had code developed for driving that allows for waypoint navigation using location feedback based on dead-reckoning data from wheel encoders and an onboard gyroscope. The dead-reckoning data is fused through Kalman filtering with position and orientation data from the OptiTrack system, which is broadcast over the WLAN from the PC connected to a WISER antenna. A Python script transforms the OptiTrack data and associates it for distribution to the robot cars.



For data collection, the robot car program was started with waypoints such that the car drove on the same path each circuit, except for the first run, during which it had to get from the starting point onto the circuit. The python script for broadcasting OptiTrack data to the network was modified to create a list of position data with WISER and OptiTrack data on the same line. At each data collection instance, data was polled from both systems in the same statement to approximate simultaneity. The WISER system requires gathering via http in json formatting as described in the WISER documentation. The python module “requests” can achieve this. Every 1000 data points (~5.5 minutes), all the data is dumped into a .csv file for further processing in MS Excel using the “text-to-columns” feature with the correct delimiters selected. An outline of the python script is included in the figure below. An identical procedure was used for collecting static data, although in this case the robot car was left sitting still at the entrance to the course, as depicted in Figure 2.5.

```
import requests
import json
import csv

starttime=time.time()
wiser_Data = ([" wiser_y(in)   wiser_x(in)   wiser_z(in)   opti_x(m)   opti_y(m)   opti_yaw(deg)"])
timeData = []
csvfile = "NickW_posData.csv"

...

if frame_counter % 25 == 0:
    wiser_data = requests.get("http://localhost:3101/wiser/api/activetag")
    formatted_data = json.loads(wiser_data.text)
    position_data = formatted_data[0]['location'] #saves xyz location data as a tuple
    wiser_Data.append([position_data, mocap_state[1][0],mocap_state[1][2],mocap_state[1][4]])
    print position_data
    print "rbid      status      x      y      yaw"

...

if frame_counter % 25000 == 0: # 25000 frames ~= 5.5 min
    with open(csvfile, "w") as output:
        writer = csv.writer(output, lineterminator='\n')
        i = 0;
        for val in wiser_Data:
            writer.writerow([val])
```

Figure 2.6: Outline of the python script used to collect data near simultaneously from the OptiTrack and WISER systems. The data is collected in a python variable which grows with every data point. After a defined number of time steps, the data is dumped into a .csv file for further post-processing.

After data collection, post processing included rotating one of the data 180 degrees about the Z axis to line up the positive X and Y axes, converting the WISER data from inches to centimeters, converting the OptiTrack data from meters to centimeters, and moving the origin of the WISER data to be coincident with that of the OptiTrack data.

## Results

The data collected are summarized in the figure captions of this section.

A relevant discussion for this section is on the difference between accuracy and precision. Accuracy refers to measurement distance relative to a known value, while precision is only a property of the measurements to each other. The collected data shows that the WISER system is more precise than it is accurate. That is to say that the WISER data is very repeatable, but it does not fully agree with data from the OptiTrack system.

The static WISER data show a mostly normal distribution about the average of the OptiTrack data. This is a desirable characteristic, as Gaussian noise is the simplest to deal with. Similarly, for the dynamic data, it appears that although the trajectories are slightly different for the WISER and OptiTrack data, the WISER data nevertheless is very consistent and easy to imagine that it is distributed normally about its mean trajectory.

It is probable that some error is present due to the setup of the WISER system. It was difficult to locate all the antennas with high accuracy and then the graphical nature of the settings initialization contributed additionally to antenna position error. An educated guess suggests that each of these was accurate to plus-or-minus one inch.

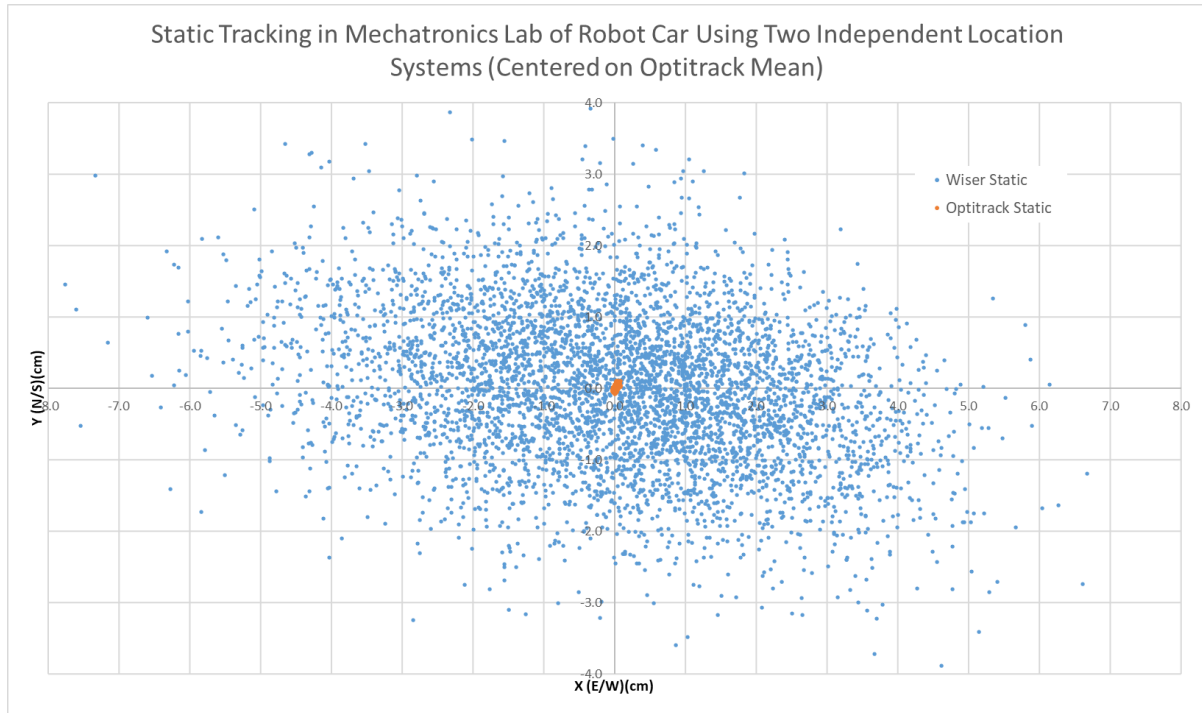


Figure 2.7: XY coordinate data of 5000 location points captured over ~25 minutes comparing the Wiser UWB location system against an OptiTrack optical motion tracking system. The data were captured at the course entrance where the car is shown in figure 5. This corresponds in Figure 8 to the cluster of outliers at the bottom of the graph. The system was at rest in a static configuration for the duration of this test. The centroids of each dataset coincide by normalizing the average of each to zero.

## Dynamic Tracking in Mechatronics Lab of Robot Car Using Two Independent Location Systems

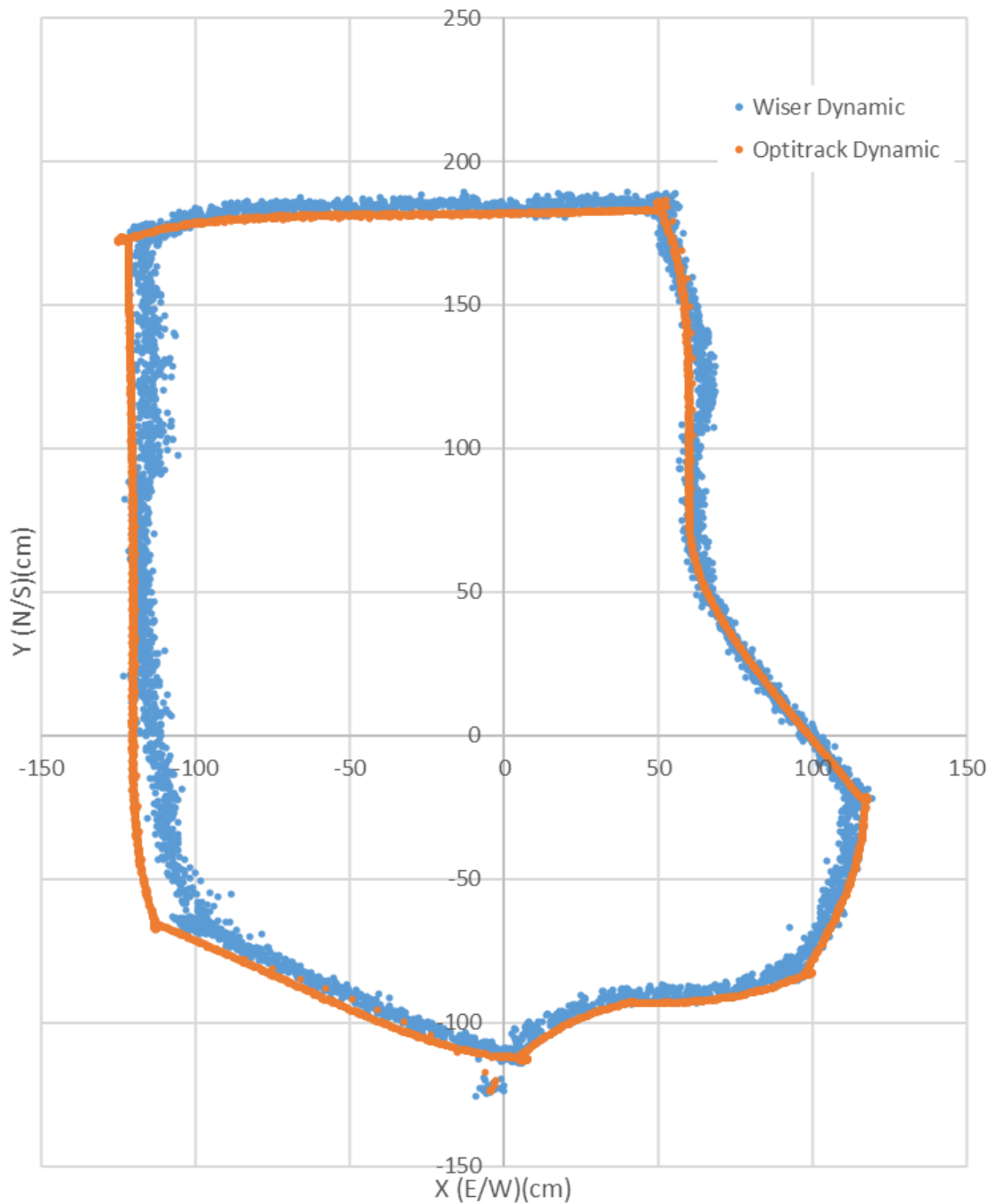


Figure 2.8: XY coordinate data of 5000 location points captured over ~28 minutes comparing the Wiser UWB location system against an OptiTrack optical motion tracking system. The robotic car was programmed to autonomously drive around the room via a series of repeating waypoints. OptiTrack data was sent to the car to be Kalman filtered with dead reckoning data from wheel encoders and a gyroscope to provide positioning feedback for the car.



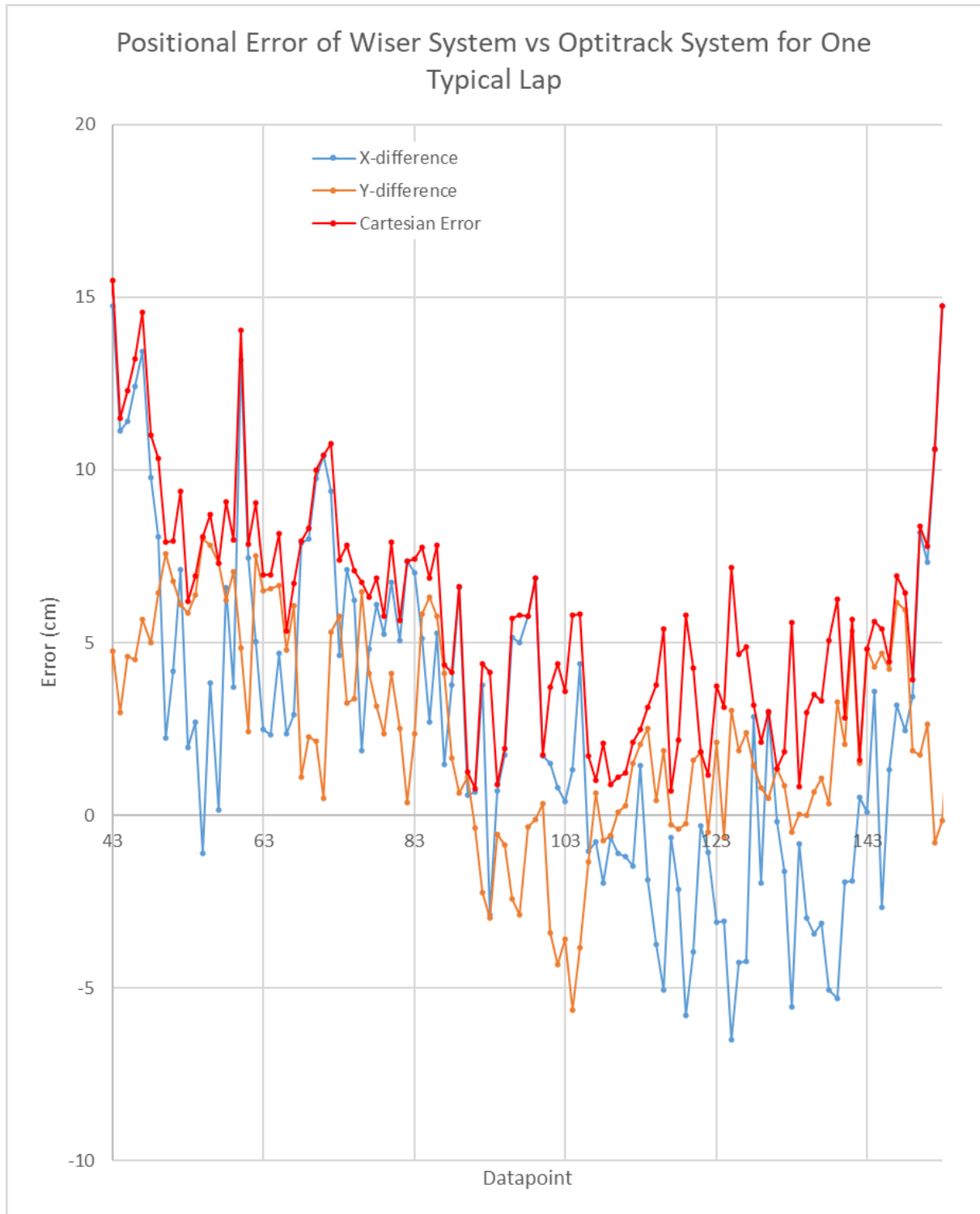


Figure 2.9: Typical positional difference of the Wiser system compared to the OptiTrack system for one circuit of the robot car's driving path. Cartesian error is the Pythagorean distance calculated between the OptiTrack and Wiser (x, y) coordinates for each position sample. It is also better known as the Euclidean metric. Beginning of the lap was chosen arbitrarily for ease of data segmentation.

Avg Cart Err(cm)	Avg X Err (cm)	Avg Y Err (cm)
6.00	2.57	2.52
Std.Dev Cart Err(cm)	Std.Dev X Err (cm)	Std.Dev Y Err (cm)
3.30	5.01	2.97

Figure 2.10: Statistics on the comparison of positional data from the Wiser system vs the OptiTrack system.  
Cart = Cartesian, Err = Error, Avg = mean, Std.Dev = standard deviation

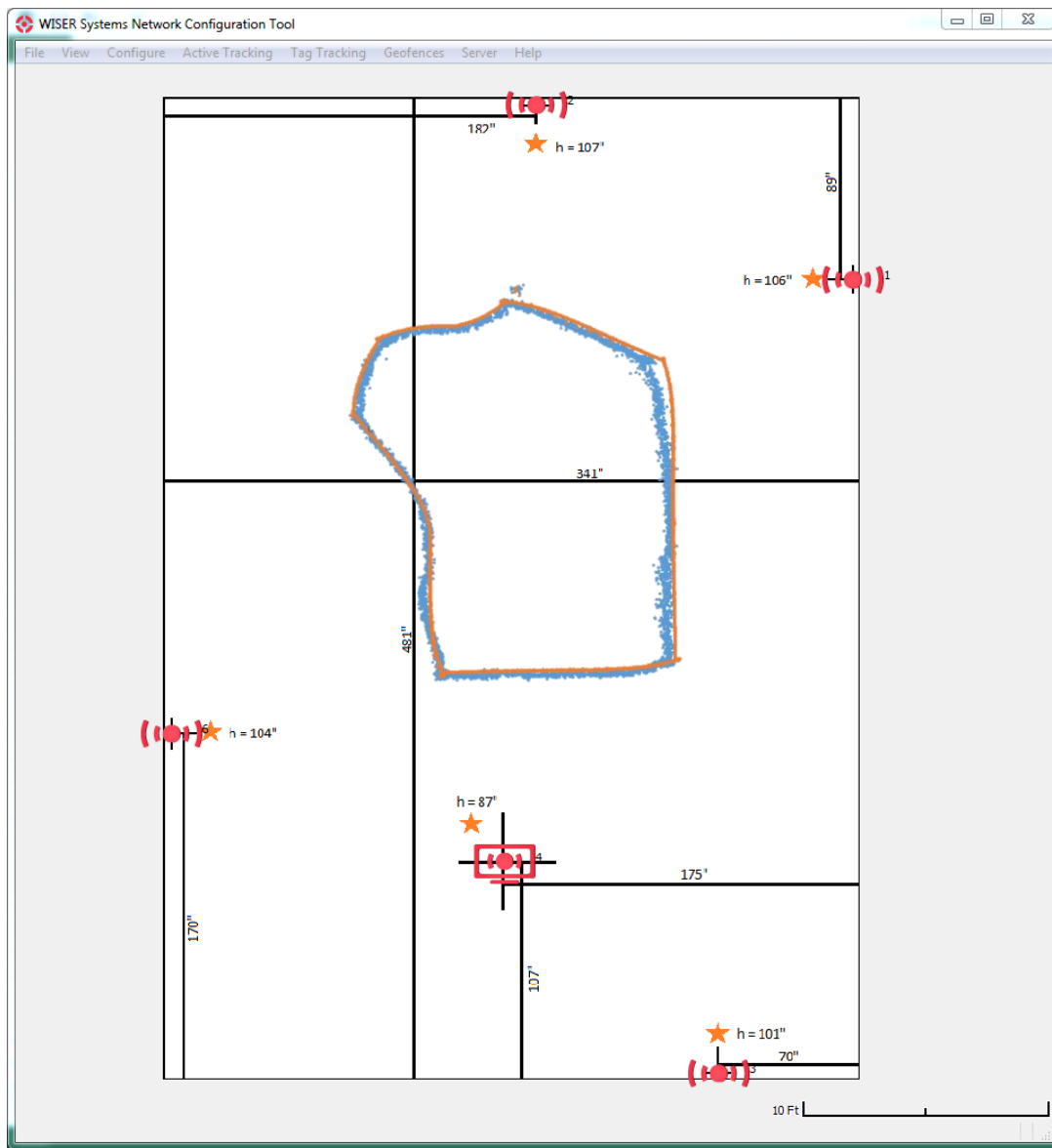


Figure 2.11: Data shown in reference to the wiser.exe layout picture in approximate scale. Note that the WISER coordinate frame and the OptiTrack coordinate frame are rotated relative to each by 180 degrees in the Z-axis.

## Recommendation

The results of this study affirm that Decawave based UWB systems are promising for use in robot localization. This is not a new result in the field and, in fact, the references in this study almost all pertain to such. The WISER system is just one implementation of the Decawave technology [87] and there exist others that show more promise in the immediate term for use in robotics applications. The Pozyx system, in particular, is ready to be used out of the box as a robot localization system with ROS.

The WISER Locator has a few weaknesses when viewed from the perspective of the robotics community. First of these is the fact that `wiser.exe` is a windows application. Modern robotics research is almost exclusively done in Linux using ROS. The reasons for this are many, but some of the most relevant are the small footprint of Linux compared to Windows, the free availability of Linux, and the open-source nature of the platform. In its current state, the WISER system would be the only application requiring windows on a robot and the data would simply be passed into Linux for use. Separately, `wiser.exe` seems to have bugs including failure to change the settings `.xml` file based on dropdown selections. These should be addressed.

In a related vein, the WISER system requires one static antenna be connected to a device running `wiser.exe`. It is not clear whether `wiser.exe` might be able to run on an embedded processor. The design of the application is heavily based around its GUI, although the settings `.xml` is what fully defines the configuration. It is unknown whether command line control would be readily implementable with the application or if a complete redesign of the software would be necessary. Currently, the way to begin active tracking is by using the dropdown menu in the GUI. WISER Systems does say that it is possible to connect the static antenna to an IP accessible gateway for remote control, although `wiser.exe` must be running somewhere. There are open questions regarding the ability of `wiser.exe` to be used embedded, however at the present moment these capabilities do not seem to be advertised or supported.

A third large inconvenience of the WISER system is perhaps one of its strengths in another market segment. The antennas are of a very awkward geometry. This makes sense when wall-mounting in a warehouse with the supplied female holders, however mounting these on a

robot proves challenging. The author would prefer something with more right angles and preferably flanges for rigid, bolted, mechanical connections. Ideally, the housing will be rated for ingress to a rating of IP67/68 or equivalent for use in continuously wet and/or dusty environments. Additionally, the transceiver's point-source location should be clearly indicated on the housing and documented relative to the bolting points or relevant reference datum.

Are there alternatives to the WISER system more applicable to robotics? Yes, there currently exist at least two superior implementations of the Decawave DW 1000 chip. There are a few development modules offered directly by Decawave [81], which have been used for research in several publications [73], [74], [75], [76]. The other popular robotics implementation seems to be the Pozyx system [82], which has been used in numerous papers implementing the Decawave chip [83], [84], [85]. Both systems are much less expensive than the WISER system as of September 2018 and provide much greater configurability to meet the needs of the robotics community.

## CHAPTER 3: Developing a ROS-Based Experimental Infrastructure

This chapter provides an explanation of the essential software tools used in the next several chapters, which are focused on statistically characterizing UWB ToA multilateration results. There are several pieces of hardware used in this research other than motion capture introduced here, but they will be introduced in the order and section in which they first appear.

### Decawave TREK1000 Development Kit

For most of the research timeline of this thesis, Decawave’s flagship product offering was the DW1000 integrated circuit chip that was touted as “the world’s first single-chip wireless transceiver based on Ultra Wideband techniques” [88]. According to the document history in its datasheet, the initial production was November 2012 [89], however the earliest mention of the chip is in a November 2013 article announcing its introduction [90]. The DW1000 chip was at the heart of all of Decawave’s products and most UWB products available until the release of the DW3000 chip around November 2020 [91], which integrates more components into the chip, lowers current consumption compared to the DW1000, and enables phase difference of arrival (PDoA) operation from a single chip [92].

Each of Decawave’s other products is designed to build development capabilities on top of its core integrated circuit offerings, as they are truly a semiconductor company [93]. These offerings vary from basic DW1000 antenna implementations, to DW1000 antenna plus IMU units, to full microprocessor implementations with enough processing power to act as an internet of things (IoT) device. The focus for this thesis is on the Decawave TREK1000 evaluation kit. This kit is a collection of four individual EVB1000 evaluation boards which have been factory calibrated to compensate for antenna delay [94]. TREK1000 is designed to allow evaluation and parameter experimentation of ToA localization using the DW1000 UWB chip. It supports a system of up to four anchors and theoretically up to eight tags, although this number is practically limited by frame interference between adjacent tag addresses, as will be discussed.

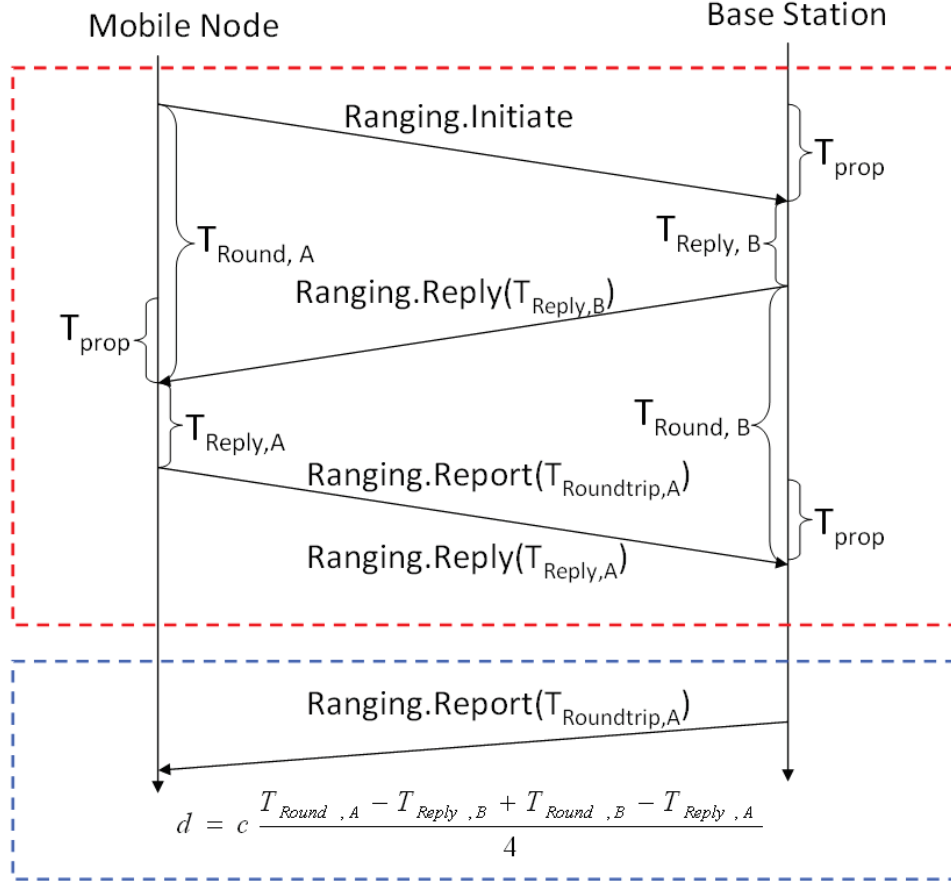


Figure 3.1: Two-way ranging scheme to enable ToF distance measurements without clock synchronization. [95] The upper dashed red messages are the necessary three to calculate ToF at the anchor. The lower dashed blue message is an additional transmission sent to the tag to inform its position exactly one period after the distance,  $d$ , has been calculated by the anchor.

TREK1000 implements two-way communication to facilitate clock synchronization in ToA measurements by utilizing an algorithm that Decawave calls two-way ranging (TWR), the scheme shown in Figure 3.1. By sending three messages between the tag and the anchor, enough information is exchanged to allow clock synchronization between the two antennas. The way this works is that first a message is prepared from the tag with a timestamp and sent to the anchor. The anchor takes time  $T_{Reply,B}$  to send a message back with the anchor timestamps of message receipt and transmittal. Once the tag receives this reply, it can calculate  $T_{Round,A}$  and send an equivalent reply, taking time  $T_{Reply,A}$ , back to the anchor, which can then calculate  $T_{Round,B}$ . These four times allow the calculation of equation (3.1), the time of flight.

$$ToF = \frac{T_{Round,A} - T_{Reply,B} + T_{Round,B} - T_{Reply,A}}{4} = T_{prop} \quad (3.1)$$

The ToF is then further used to calculate distance between the tag and anchor,  $d$ , traveled using the speed of light,  $c$ , in equation (3.2).

$$d = c(ToF) \quad (3.2)$$

An important implementation detail to note is that “in the TREK1000, where the tag is continually ranging, the resultant ToF value calculated by each anchor is returned to the tag, during the next ranging exchange” [96]. This means that the tag reports the previous timestep’s ranging values during the current timestep and this must be accounted for in analysis. The tag performs this TWR with each of the available anchors and performs an implementation of the trilateration algorithm, equation (1.5), once per timestep.

The previous explanation assumes direct antenna-to-antenna signal transmission, but in reality, the signal must also travel along a short distance of copper between the antenna and DW1000 unit. This additional electrical transit length introduces a non-negligible amount of time delay into the system, unique to each unit, that reduces the EVB1000 module accuracy by 25cm beyond the design specifications [94]. Figure 3.2 illustrates this additional delay in the ToF diagram and shows that each of the tag and anchor board delays occur three times in the TWR process. Each of the TREK1000 units has come from the factory uniquely calibrated to account for this delay, but EVB1000 units must be calibrated by the consumer before use.

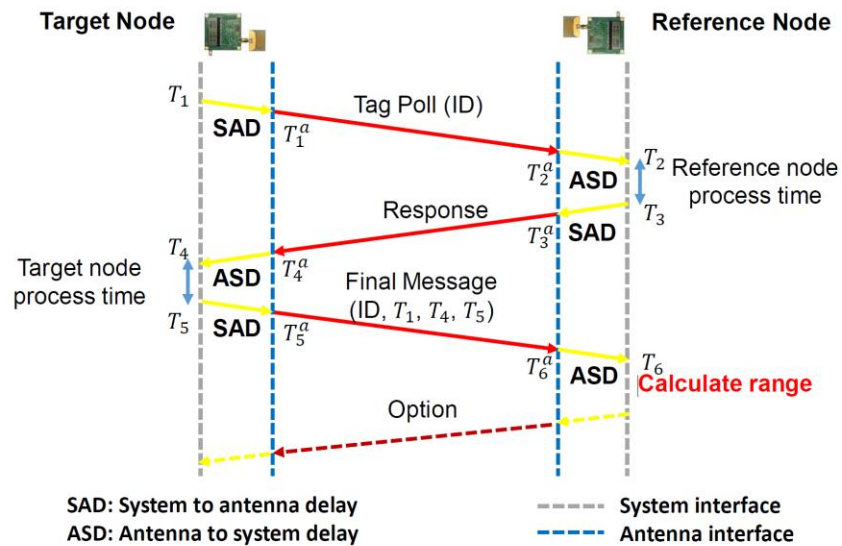


Figure 3.2: Antenna delay must be accounted for when calculating ToF. [97]

If one examines the TREK1000 source code, the specific trilateration algorithm implementation can be observed. It uses iterative geometric reasoning to reduce the possible locus of 3D positions from a sphere to a circle to two points, using distance information from anchors zero, one, and two, respectively as shown in Figure 3.3. The fourth anchor is optionally used to deduce height from the two solutions, otherwise the lower one is selected if this feature is turned off. This is not the most robust algorithm as it fails if one of the first three anchors is missing and weighs each differently in its calculations. Other multilateration algorithms ([98], [99], [100], [101], [102]) could be used in lieu of the one from Decawave to add more anchors and possibly increase system accuracy, but that task is left for future research.

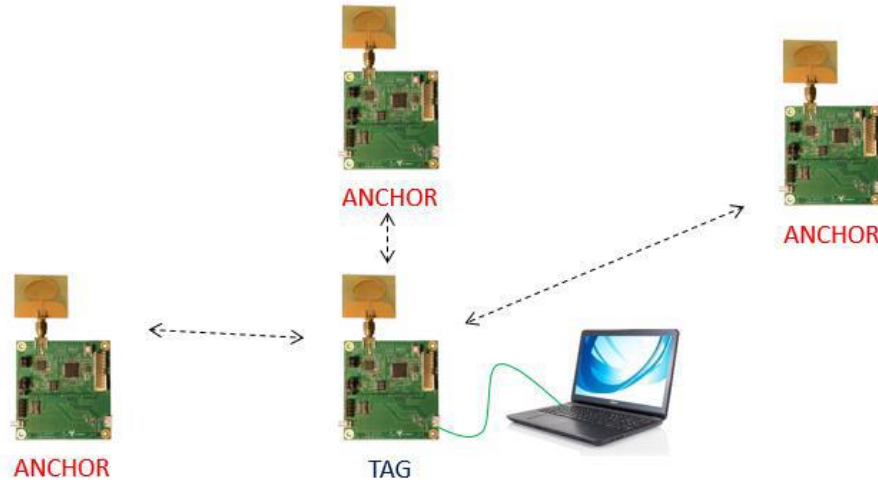


Figure 3.3: Trilateration is implemented using the TREK1000 software development kit from Decawave. [103]

Each of the TREK1000 modules comes flashed with identical serial information from Decawave. This creates an impossibility of uniquely determining connections in a robotic application. The typical solution is to utilize udev rules in Linux to differentiate between hardware, however this requires flashing new information to identify the hardware with. The ST-LINK/V2 flashing tool and debugger shown in Figure 3.4 is used to connect to the boards and reflash them. There is a line in the source code that can be replaced with the unique serial number attached to each board as a sticker. This then allows for unambiguous identification by the computer without using an additional USB hub or requiring sequential connection of units.

TREK1000 supports UWB channels 2 and 5, along with data rates of 110k and 6.8M. All experiments in this thesis were carried out on channel 2 at 110k to maximize system range.





Figure 3.4: An ST-LINK/V2 module used to flash the TREK1000 boards.

## ROS

“The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms” [104]. Before the existence of ROS, academic robotics research mainly involved writing software infrastructure to enable newly acquired robot hardware simply to operate as intended. This took considerable amounts of time and re-implementation of code in each individual lab before novel research and experimentation could even commence. ROS aimed to change this by creating a common, open-source software infrastructure that could be used to get a robot up and running quickly without having to develop from scratch for each new robot [105].

While there is much that could be said and has been said about ROS, its history, and its multitude of capabilities, this thesis only utilizes a small subset of the full package. Although RViz was used at times for visualization, the most utility has been gained from using the `ros_comm` packages `msg`, `message_filters`, `rosbag`, and others which generally make up the core of the ROS middleware [106]. ROS was also helpful in providing a working C++ development platform to start coding quickly. The reader is left to research the vast and rich ecosystem of information that is ROS on their own. Suffice it to say that ROS is the standard for robots across academia, industry, and military at the time of this writing.

## ROS Decawave TREK1000 Driver

To fully utilize the power of ROS, a custom localization driver was written for the Decawave TREK1000 system. Conceptually, there are several steps that the localization driver must

execute, including reading hardware serial data, converting this into ROS compatible information, performing trilateration on the data, and finally publishing the calculated pose on a ROS topic. The overall structure of the custom TREK1000 is visualized in Figure 3.5, which shows the ROS rqt\_graph structure of the encapsulating namespace, `uwb`. Typical drivers would incorporate everything between serial input and pose output into a single ROS node. This is a more efficient implementation for real-time hardware, however ROS1 is not real-time capable, the TREK1000 is only a demonstration kit, and this thesis is focused on in-depth data analysis. For these reasons, the TREK1000 driver was broken into three separate ROS nodes that share information using ROS’s messaging paradigm. This segmentation allows rosbag to collect experimental data from each node for later additional analysis and facilitates easily rerunning experiments on the same data using different algorithms in the future. It also allows multiple instances of the same node to run in parallel, with different configurations based on system hardware requirements. This characteristic will be used to run single-tag and dual-tag localization using the same code with only a parameter variation in the launch file.

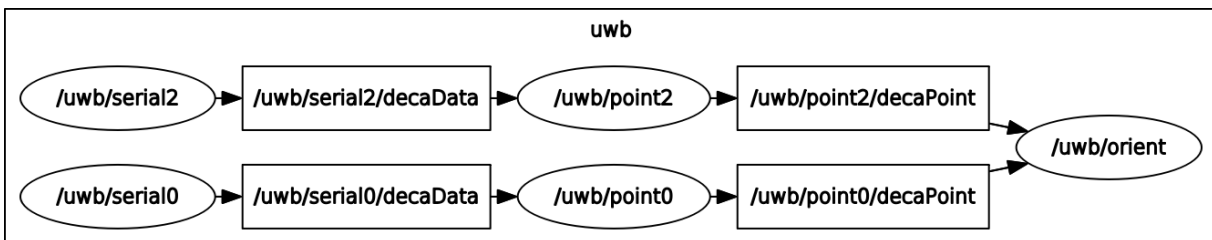


Figure 3.5: Structure of the ROS TREK1000 driver shown for dual-tag localization. Single-tag localization uses only one of the chains and does not include the orientation node.

The first of the conceptual functions is message transfer between the hardware peripheral and the computer running the driver. For the TREK1000 system, this is accomplished using a serial connection over USB. Once the hardware units are setup and running, data automatically streams to the serial port. A ROS node was written specifically to continuously poll the serial port for new data, parse this serial data based on the message structure, and reserialize it into a custom ROS message that is published from the node for the rest of the system to see. This serial node provides raw ranging data to be utilized in a multilateration algorithm. The serial node has parameters in the launch file that allow serial ports to be defined based on their names. In combination with udev rules, this allows multiple instances of the same hardware to

be uniquely identified in the ROS system, independent of physical serial port connection or order in which the hardware is connected, as Ubuntu assigns generic serial port names based on the order in which they are connected.

The second node in the driver chain consumes raw distance data from the serial node, performs multilateration on the ranges, and returns a position estimation relative to the configured UWB anchor locations. The position is returned in a standard ROS geometry\_msgs type [107]. The information contained is that of a PointStamped message with point position, sequence, timestamp, and frame\_id data. Pose messages are also possible to publish if needed, but the orientation data will necessarily be static.

A third and final orientation node is optional, depending on the specific hardware setup. If position tracking is all that is desired, then just the serial and multilateration nodes are required. However, by utilizing two UWB tags on a robot, it is possible to draw a line between them that defines the robot as a rigid body. From the individual tag positions, one can define the translational and rotational offsets from the robot base\_link frame and calculate full pose information from the two tags. Equation (3.3) and Figure 3.6 show how orientation can be calculated from the two UWB tag locations within the map frame. Additionally, if the tags are symmetric such that a line drawn connecting them passes through the robot origin, then the robot origin can be tracked simply by averaging the position readings of each of the two tags. This simplifies the transformation math, but it is also possible to transform each tag location back to the robot center and average positions there without requiring the tags be placed on a line intersecting the robot origin.

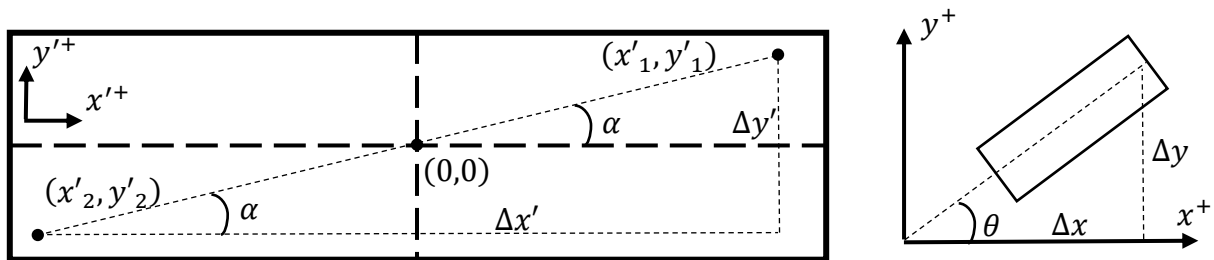


Figure 3.6: Calculation of the robot heading angle, theta, from UWB point locations. The robot heading angle is calculated by considering the orientation of the line created between the two UWB points and offsetting the constant angle, alpha, that they are already mounted on the robot with. Left: tags installed on the robot. Right: robot within the environment.

The further apart the two tags can be spaced, the more immune orientation calculation will be to noise. Equation (3.4) shows conceptually why this is the case. The further apart in the  $y$ -axis that the tags are, the smaller the derivative of  $\arctan()$  will be, which means that  $\theta$  will be less sensitive to noise. In reality,  $\text{atan2}()$  is used in the calculation and any greater distance between the tags will decrease noise in the angle calculation.

$$\theta = \text{atan2}\left(\frac{\Delta y}{\Delta x}\right) - \alpha \quad (3.3)$$

$$\frac{d}{dx} \arctan(x) = \frac{1}{1+x^2}, \quad x = \frac{\Delta y}{\Delta x} \quad (3.4)$$

Currently, only heading angle is being calculated and published from the orientation node, but it is possible to modify the code to calculate all three pose angles, if needed. The tag positions relative to the robot frame are currently written into the launch file, but the ROS package `tf2` could be used with tag positions recorded in the robot URDF for a more permanent solution and to facilitate asymmetric tag mounting about the robot origin.

All the experiments in the rest of this thesis relied on this ROS TREK1000 driver code or a similar refrain on it for the Trimble SX-10 and Decawave PDoA drivers. Please reference the appendix for the complete source code utilized in this work.

### Motion Capture Systems

Motion capture systems (MoCap) utilize infrared light and retroreflectors to locate a collection of centroids in multiple camera frames of reference for comparison to perform triangulation. The cameras have an array of infrared light LED that are pulsed into the MoCap area to reflect off spherical retroreflectors and return to the camera lens. See Figure 3.7 for a glimpse of the infrared light pulses otherwise invisible to the human eye that the MoCap cameras detect.



*Figure 3.7: MoCap infrared light output captured by cellphone camera. The rapidly pulsing nature of the infrared LEDs is captured in the picture on the left.*

Motion capture (MoCap) systems are used to measure robot movement accurately and precisely. The current system in use from Vicon has 12 Vero cameras for a maximum tracking volume of about 30x50x10ft. Vicon has a good explanation of what MoCap is on its website [108]. In essence, multiple cameras are calibrated together into a system which can triangulate points in space based on their appearance across multiple simultaneous images. This allows the system to measure pose (position and orientation) and then infer twist (velocity and angular velocity) and acceleration (linear and rotational) in practically real-time.

The accuracy of a Vicon MoCap system has position error measured on the order of millimeters and a maximum refresh rate of 330Hz. The level of accuracy and the speed at which data can be collected allows us to assume MoCap data as ground truth (actual state measurements) as compared to other sensors which provide less accurate estimates of the true state. Note that MoCap data is also just an estimate of the true state, but it is typically more accurate than any other sensor by at least an order of magnitude. The system also provides absolute measurements (relative to the MoCap system) as opposed to the relative measurements that sensors such as IMUs and wheel encoders provide. This provides a method for evaluating relative (and other absolute) sensor performance within the map frame.

MoCap is useful to characterize the measurement errors which other sensors introduce and ultimately use that information to tune the localization stack of robots. Most robots use a Kalman filter to fuse together multiple sources of sensor data to provide an estimate of the state of the robot and the accuracy of this filter is increased when it has better information on the accuracy level of each individual sensor measurement. MoCap system is useful for tuning because it is impractical to expect a MoCap system anywhere other than a controlled laboratory environment. It is possible, however, to feed MoCap measurements directly to a robot to develop and debug control algorithms while isolated from other sensor noise. This is what many research groups do to get their drones to do flips, somersaults, and other impressive maneuvers which would be difficult to execute precisely without MoCap data.

Generally, Vicon hardware and software is simple and intuitive to use, but there are some nuances to utilizing it correctly. When creating an object using multiple retroreflectors, it is best practice to mount the spheres on surfaces which can be seen by as many cameras simultaneously as possible, otherwise the view may be obfuscated depending on robot heading. Various pictures throughout this document will show retroreflector placement and details discussed in the relevant sections.

The Vicon MoCap system was used at a constant refresh rate of 100Hz throughout all experiments in this thesis. This was much faster than any of the other sensors characterized and fast relative to the robot speeds used in data collection. The Vicon Tracker software [109] enables system operation in engineering specific applications, such as robotics. To interface the Vicon Tracker software with ROS, the `vrpn_client_ros` package [110] was utilized. This is a simple to use software package that reads object data from the virtual reality peripheral network (VRPN) server built into Vicon Tracker in real-time via an Ethernet connection. The documentation is succinct and the only setting one need worry about is “`use_server_time`”, which the next section on clock synchronization will discuss in detail.

### [Clock Synchronization \[111\]](#)

ROS depends on clocks to accurately timestamp messages. The key assumption of accurate and synchronized clocks underlies much of ROS’s basic functionality. Were there no timestamps

involved, rosbag would have no way of knowing when events occurred relative to each other for playback. One alternative solution to time might be cycle counts, and this is the approach typically used by monolithic real-time control systems. There would be a master clock with one or more deterministic control loops running with enough processing overhead to guarantee a high rate of cycle completion even when considering all interrupts that could occur.

ROS enables systems that are not so restricted. For example, distributed systems of multiple robots and computers each with their own internal clocks. The issue that arises with such a system is how to keep all the clocks in sync to read the same time. This issue arises not just once at the beginning of experiment time, but periodically since different clocks tick at different rates. This is especially true for the quartz oscillators in most computers. Figure 3.8 shows an illustration of clock drift that occurs when clocks tick at different rates.

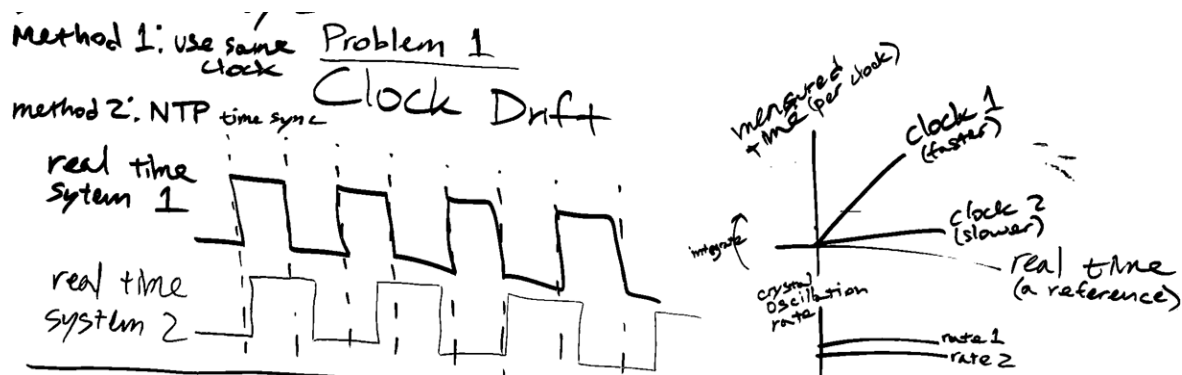


Figure 3.8: An illustration of the problem of clock drift. Individual clocks tick at slightly different rates. Even if two clocks are synchronized to be the same at a point in time, their differing crystal oscillator rates will cause them to diverge over time. This phenomenon is known as clock drift.

The solution to this issue is a method of clock synchronization between multiple clocks [112]. A full discussion of how to achieve this is out of the scope of this document, but three feasible solutions for robotics are:

1. Connecting GPS/satellite/radio enabled clock synchronizing hardware to each device.
2. Utilizing a local time server as a source for network clock synchronization.
3. Using a single hardware clock across multiple machines.

The next section will discuss attempts at solution 2 to synchronize two computers before acknowledging that solution 3 is better suited for importing Vicon MoCap data into ROS.

## Synchronizing Windows 10 Time to an Ubuntu Linux Server

Returning to the MoCap section's discussion, Vicon Tracker software is only supported on Microsoft Windows 10 [109]. There is available an SDK that runs on 64-bit Linux, however it also requires an instance of Windows software running [113]. Thus, ROS and Vicon software must necessarily be running on two separate machines. The `vrpn_client_ros` configuration parameter "use\_server\_time" tells ROS whether to timestamp data with the VRPN server time from the Windows 10 machine ('true') or with the time from the local ROS clock ('false'). If using the VRPN server clock for timestamping ROS messages, then it is necessary to synchronize clocks between the machines to ensure time-consistency between data sources, as the rest of the sensors are timestamped using the Ubuntu machine's clock. A complicating factor in this scenario for mobile robots is that a wireless connection is desired for the onboard computer.

The rest of this section will be a discussion of the practical results obtained when attempting to utilize Ubuntu Linux machines as a time server source for clock synchronization with a Windows 10 machine. Windows 10 natively implements the Network Time Protocol (NTP) [114] and claims to support clock synchronization on the order of 1ms accuracy [115]. Ubuntu Linux also natively supports NTP and can provide a time server for Windows 10 to sync to using Chrony [116]. Chrony simply must be told it is running a low stratum clock and the Windows time service pointed at the network address of the local Linux machine. Figure 3.9 shows the slew rate adjustment that occurs over time and the resulting clock synchronization between two hardwired machines. Note that this process took about 30 minutes to stabilize to within 1ms.

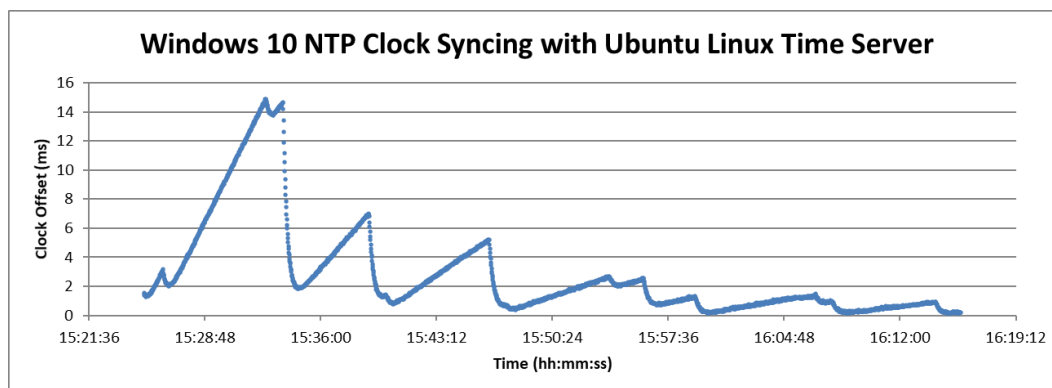


Figure 3.9: Time syncing a Windows 10 laptop to a hardwired Ubuntu Linux laptop acting as an NTP time server. The data shows that it took approximately 30min for the clock offset to sync and stabilize down to within 1ms of error.



If we zoom in on the stabilized data, we get Figure 3.11, which shows that the clocks are synced to within  $\pm 1\text{ms}$  of each other as stated possible by Microsoft. The process showed great promise, however results were inconsistent.

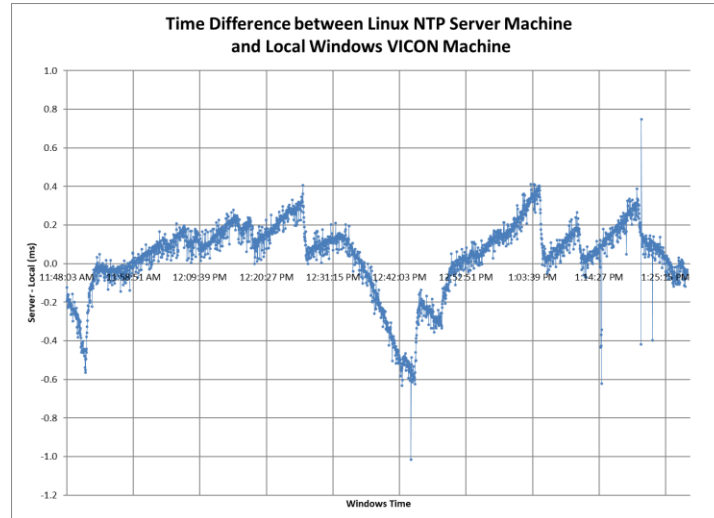


Figure 3.10: Stabilized clock synchronization offset between hardwired Windows 10 and Ubuntu Linux time server using NTP.

Figure 3.11 below shows another set of data with the same physical setup. It appears that the clocks are not fully synced yet, but offset is decreasing over time. Latency is constant over time, as would be expected from a hardwired connection. The key takeaway from these experiments is that it takes time after startup for a computer clock to stabilize and accurately synchronize with a time server.

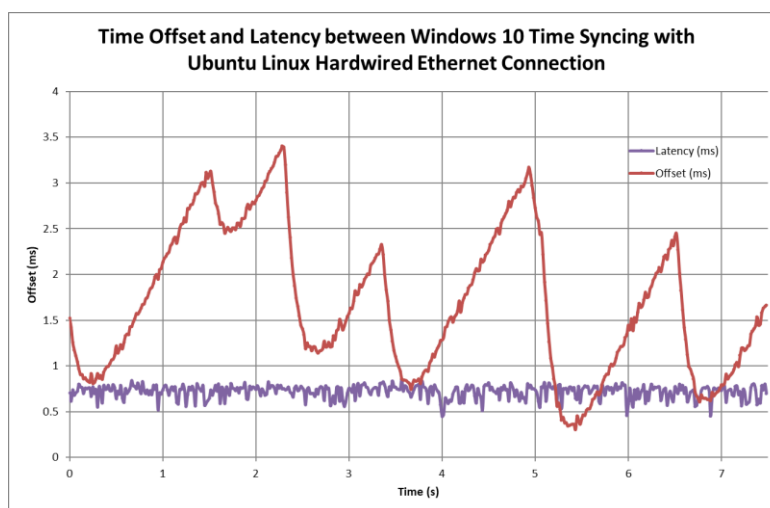


Figure 3.11: Clock synchronization offset and latency for a hardwired Windows 10 client to Ubuntu Linux NTP server connection. Clock offset is decreasing oscillation magnitude over time and latency is near a constant value of 0.75ms.

Next, these results were ported over to a Clearpath Jackal robot that was wirelessly communicating via a Microhard radio connection to a base station with a receiver Microhard radio that was hardwired to the Windows 10 machine. Many attempts were made to replicate the results of the hardwired network clock synchronization, but they ultimately failed. The wireless connection surely did not help, but one of the root causes of the failure to clock sync using NTP was the limited battery power that the Jackal had. This coupled with the need to frequently restart meant that there was not much more than 30 minutes available to run experiments, let alone wait for clock synchronization to stabilize.

As a fix, a PowerShell script was written for the Windows 10 machine to force manual clock synchronization to the Jackal computer every five seconds. This resulted in an overall very accurately synced clock, but with time jumps required to achieve that accuracy. Time jumps are not something that NTP does, as it can be undesirable to have events occur in the present but timestamped before an event that occurred prior to the jump. For this reason, NTP varies the rate of the clock, but always ensures that it is monotonically increasing. Time jumps are not such an issue with ROS, especially if jumps are bounded within a small enough time difference. Figure 3.12 shows the results of implementing this manual resync over the wireless network.

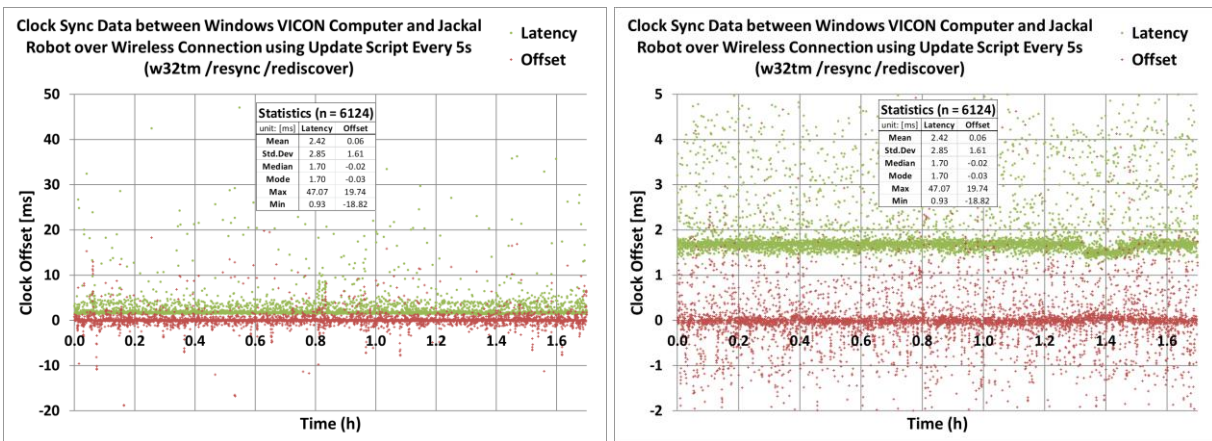


Figure 3.12: Left: Clock synchronization latency and offset data between a wirelessly connected Windows 10 NTP client and a mobile Clearpath Jackal robot NTP server. Central values of latency and offset look good, but they contain extreme outliers that add noise to sensor data. Data was taken once per second using the built-in w32tm logging tool. Right: Zoomed in view of data in the plot on the left.

Results indicate on average overall acceptable performance, but with unacceptable outliers. Maximum clock offsets were  $\pm 20$ ms, with many outliers outside of  $\pm 10$ ms bounds. For the

Vicon system operating at 100Hz, 10ms is the spacing between data points and simply will not do for collecting ground truth data. Figure 3.13 shows that the data is not normally distributed, but rather has a strong central tendency with very long symmetric tails. Reasons for the outliers could include forced clock syncing during high latency interactions due to the wireless radio connection or high CPU demand cycles that cause erroneous calculations of clock offset.

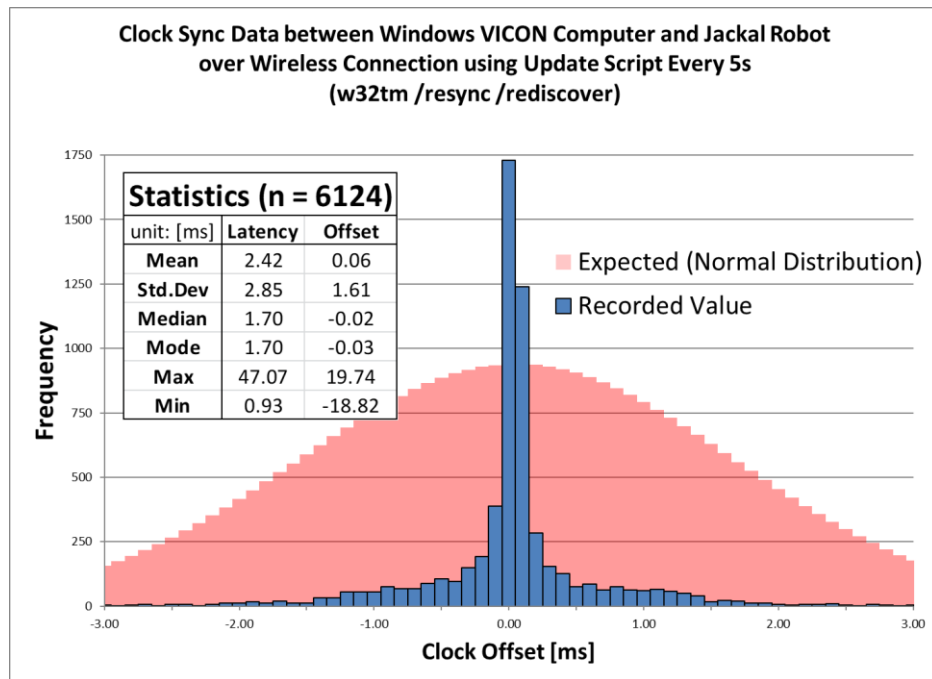


Figure 3.13: Histogram of clock offset frequencies between a wirelessly connected Windows 10 NTP client and a mobile Clearpath Jackal robot NTP server. Data is strongly centrally distributed with acceptable statistical metrics for mean and standard deviation, however the distribution is not Gaussian. There are extremely long symmetric tails that prevent the distribution from being considered normal.

The conclusion of this investigation was giving up on clock synchronization between machines in favor of using a single clock. The Vicon system must have a hardwired network connection into the same computer as either an UWB anchor or tag to accomplish this. Collecting data from an anchor is easier, as the collection computer can be stationary, and the mobile robot can be fully wireless. The UWB anchor also returns position measurement in the same frame from an anchor. If using multiple tags for orientation, the mobile robot must be used for data collection from two tags, according to the current driver software. This configuration also makes more sense for real implementation of the UWB system on a robot. However, this means connecting an Ethernet cable to the mobile robot to carry Vicon data. This is possible with the correct setup to hang a cable from the ceiling but is burdensome in larger spaces if not

simply making an experimental drive. Nevertheless, it is recommended to either hardwire a single clock or investigate other clock syncing hardware connections rather than try to synchronize two regular computer clocks for short duration experiments.

### ROS Approximate Time Message Filtering

The second time-based problem that arises in ROS is the fact that it is not real-time. Figure 3.14 shows a schematic illustration of this issue. The problem that arises for statistical sensor comparison against ground truth data is that one cannot count on correspondence between data points just by looking at their sequence count as could be done in a real-time system. Message timestamps must be compared directly to determine simultaneity, however no two messages will ever have the exact same timestamp, unless there are multiple process threads running in parallel.

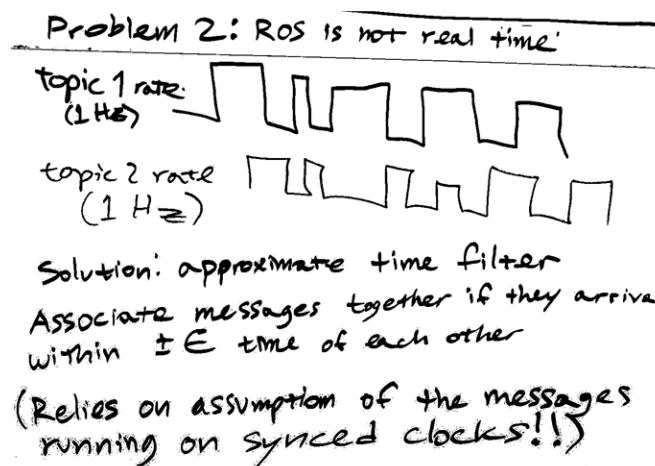


Figure 3.14: Two ROS topics running at the same average published rate have no guarantee of deterministic periodicity. Given data from two sensors that is created/pollled at the same instant in the real-world, ROS may offset their timestamps due to competing cycle time constraints. Message queues may fill and empty sporadically leading to a loss in one-to-one comparison between sensors that would ideally sample simultaneously.

The solution to this problem is given by utilizing the ROS `message_filters` package `ApproximateTime` policy [117]. This message filter policy takes in up to nine separate ROS messages as input and executes a callback function of choosing when each of the topics approximately matches up in time, according to the algorithm described in [118]. To facilitate direct comparison of sensor and ground truth data in MATLAB post-processing, filters were written to consume the two ROS message topics and publish them in the callback as new, filtered topic names. The basic structure of this filter is shown in Figure 3.15, although it is

usually desirable to publish both topics again independently on new topics instead of combining them into a single new topic with a custom message.

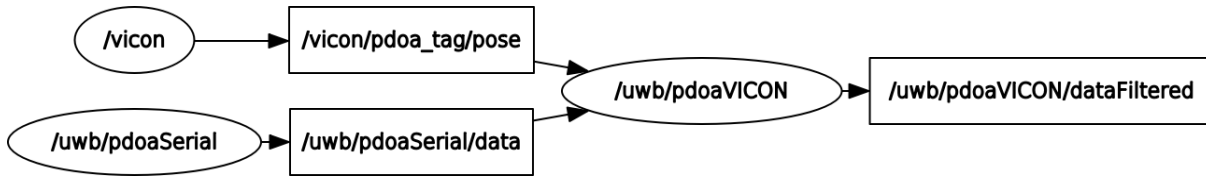


Figure 3.15: Basic structure of all ROS approximate time filters. Each sensor data point is matched up approximately in time with a ground truth data point from the Vicon MoCap system and both points are republished onto new topics.

This approximate time filtering paradigm allows time to drop out of consideration when doing statistical analysis. When implemented correctly, each filtered topic should have the same number of data points present, each index of which are estimates of quantities taken as close to the same time as possible without a real-time system with interrupts. The 100Hz MoCap ground truth data was much denser in time than any other sensor considered in this thesis (UWB was at  $\sim 3.5\text{Hz}$ , Trimble SX-10 at  $\sim 5\text{Hz}$ , and UWB PDoA at  $\sim 10\text{Hz}$ , on average), so for all intents and purposes of the following experiments, ground truth data was taken simultaneously as sensor data. This data filtering in ROS makes analysis much easier and allows vector elements to be compared one-to-one in MATLAB. Still, the original data remains and can be plotted for better visualization of the system reality, while point-matching facilitates statistical comparisons and point-cloud transforms, as will be discussed in a later section.

## MATLAB

Various MATLAB toolboxes were used throughout the data analysis portion of this thesis. The ROS Toolbox was used prominently for parsing bag files full of data, however there are other free and open-source methods for accomplishing this. Some of the conversion functions belong to various toolboxes, but they could easily be written as calculations. Some machine learning was done in MATLAB, but this is easily done in Python. In summary, MATLAB was used in this thesis out of convenience for data visualization, not out of necessity. The author also spent some time making 3D PDF plots for data communication, but these are generally not accepted for publishing. See [119] and especially [120] for how to generate 3D PDF plots from MATLAB.



## CHAPTER 4: Small-Scale Decawave TREK1000 UWB Testing

This chapter discusses the detailed setup of the Decawave TREK1000 system, configuration of the Vicon MoCap system, and experimental results obtained in a typical lab-sized room referred to as the ECR. This room provided enough space for a small deployment of the 12-camera Vicon Vero system into a MoCap arena for the Clearpath Jackal, a small, unmanned ground vehicle (UGV) [121], to drive around in. Figure 4.1 below shows the layout of the Vicon arena with UWB anchors installed around the ECR.



Figure 4.1: Vicon arena of size 6.55m x 4.37m x 2m (258" x 172") (21.5' x 14.3'). Lower figure is a panorama of the same room taken from inside the Vicon MoCap arena. UWB anchors are circled in red.

### Trimble S3 Total Station

In Chapter 2, three possible measurement methods were cited for initializing positions of the UWB anchors. None of the methods was entirely accurate and they all relied on the assumption of orthogonality between the walls of the room. A better way to find anchor location would be from a system without the requirements of environmental constraints. The fields of surveying and metrology have tools to do just this. A Trimble S3 Servo Total Station [122] was utilized to initialize UWB anchor positions for all the experiments, see Figure 4.2.



*Figure 4.2: Trimble S3 Servo Total Station with integrated TSC3 controller.*

A “total station is a surveying equipment combination of Electromagnetic Distance Measuring Instrument and electronic theodolite. It is also integrated with microprocessor, electronic data collector and storage system. The instrument can be used to measure horizontal and vertical angles as well as sloping distance of object to the instrument” [123]. The system is first leveled to the local gravity vector and uses this as a datum for vertical angle measurement [124]. Horizontal angles are relative to the chosen direction of North and must be shot in by the user or accepted as the default datum initialized on startup. In brief, a total station measures in spherical coordinates and can convert those into any other useful coordinates, including cartesian coordinates. The Trimble S3 was utilized to provide accurate UWB anchor positions once affixed to the walls of the ECR by measuring the locations of their antenna centers. The results of these measurements are shown in Figure 4.3.

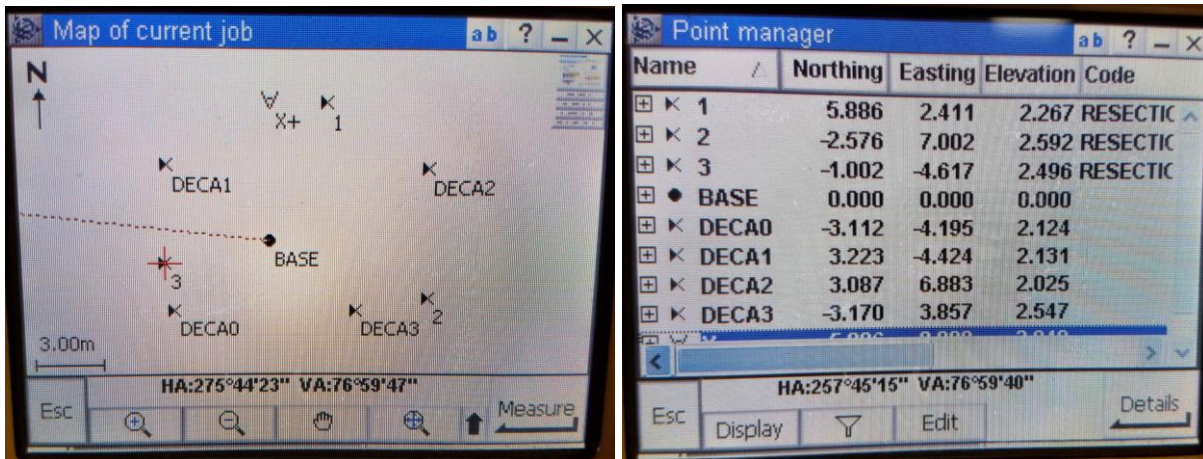


Figure 4.3: Total station calibration of the UWB anchors in the small lab space. Coordinates are returned in the NEU frame, but this is a left-handed coordinate frame. X+ in the screen capture on the left is pointing in the physical direction of East. To transform from the NEZ to cartesian coordinate frame, northing is taken as positive X, easting is taken as negative Y, and elevation is taken as positive Z. The UWB anchor coordinates can then be read off directly as (X,-Y,Z). Units are in meters.

Important to note is that total stations typically return coordinates using a northing, easting, elevation convention abbreviated as NEZ that is equivalent to the north, east, up (NEU) frame. This is a left-handed coordinate frame and of limited use when dealing with robotics. More useful is the east, north, up (ENU) frame that is conventionally used in ROS [125]. If  $ENU \leftrightarrow XYZ$ , then  $NEU \leftrightarrow YXZ$  and simply swapping the first two coordinates will transform between the frames. Figure 4.4 shows the total station angular axis conventions for horizontal angle (HA) and vertical angle (VA) measurement that, along with electronic distance measurement (EDM), lead to a left-handed coordinate system.

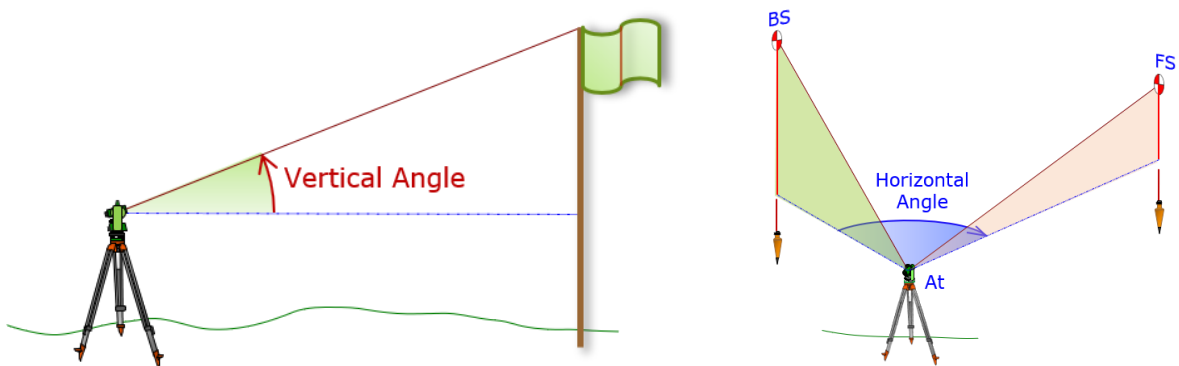


Figure 4.4: Total station coordinate frame conventions. The left image shows a positive vertical angle (VA), measured upwards from zero at level with gravity. The right image shows positive horizontal angle (HA), measured rightwards from a datum direction, such as true North. Distance is measured positive radially from the center of the total station, at the intersection of the vertical and horizontal angle axes. Due to the HA convention, these positive quantities define a left-handed coordinate system that must be accounted for in calculations. Left, (VA), source: [126]. Right, (HA), source: [127]



## Motion Capture System and Object Setup

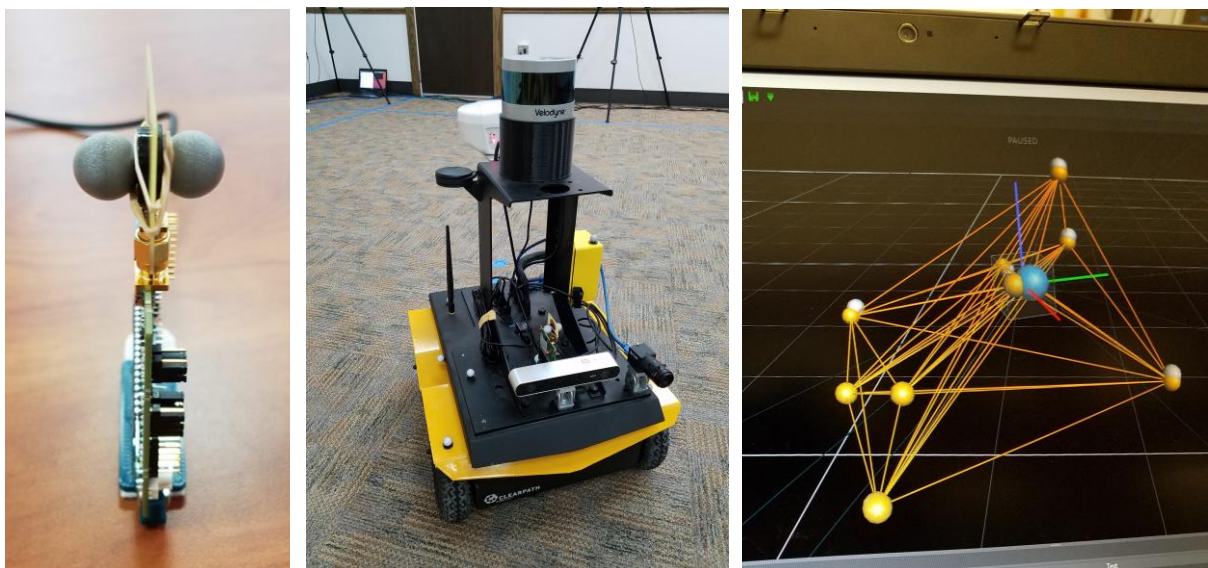
Figure 4.5 shows and explains the environmental controls used to ensure that the UWB system and the MoCap system coordinate frames remained aligned, to make data taken from each system was directly comparable. Great care was taken to make system setup repeatable, but tape eventually turned out to be not the most robust solution on a floor with multiple-wheeled differential drive robots slipping their wheels to turn. Each time the MoCap system is turned on, the best practice is to recalibrate the camera positions, just in case one of the camera tripods has been bumped.



*Figure 4.5: Masking tape placed inside of the Vicon MoCap arena to align the system with the UWB frame. In the picture to the left, the bottom right piece of tape was used to mark the origin of the total station setup, directly at the intersection of four floor carpet squares. The total station vertical angle servo was then actuated back and forth to ensure that the sight laser followed along the seam of the floor squares. The top center piece of tape, on the edge of the center spotlight, was in the path of the laser beam during this process. A dot was placed in permanent marker coincident with the laser sight and this was used to define the total station northing (X+) before shooting in the UWB anchors. The picture on the right shows how more pieces of masking tape were added to align the Vicon Active Wand used for setting the system origin. The wand origin was located by moving a retroreflective sphere over the wand until it was on the origin in Vicon Tracker and then was marked with (0,0)+ on tape. The wand origin was made coincident with the total station origin by eye and then the wand was rotated to align the positive X and Y axes with the floor. More masking tape was placed to pierce holes in using the wand's leveling screws and centering point and to draw marks on indicating how to align the wand again.*

A more robust, if less deterministic, system will be described in the next chapter for aligning data between the UWB and Vicon frames. Ultimately, relying on system frame alignment is not a feasible option for mobile robots in ad-hoc environments. A better solution is to develop an alignment procedure that has the robot drive in a straight line for several meters through the environment and use the assumption of straight-line travel to calculate coordinate transformations between various sensor frames.

Each object that is to be tracked by the MoCap system needs to be initialized by the user to set its local coordinate frame origin and orientation. Figure 4.6 shows this process in action and describes it for a single UWB tag on the Jackal robot. Single-point tracking does not require any MoCap object frame alignment, only that the origin is coincident with the sensor element that position data is coming from. No matter how the robot translates or rotates, the UWB tag point will remain aligned with the MoCap frame origin. If performing rigid body motion tracking on robots, then the object's X-axis should be positive from the rotation center of the robot towards the front and the positive Z-axis should be directed upwards against gravity. This would be the case, for instance, in Figure 4.7 on the next page, which shows a fully outfitted Jackal robot with dual UWB tags to test the orientation node. In this case, the second UWB tag is in the same plane as the first and a single retroreflector can be affixed to select and snap Y-axis rotation against in the Vicon Tracker software. The second tag address was set to be 2, compared with the first tag address of 0, because there seemed to be UWB frame interference using 1 as a directly adjacent address.



*Figure 4.6: Method used to set the MoCap origin to be coincident with the UWB tag. Double-sided tape, rubber bands, or another attachment option can be used to affix two retroreflective spheres symmetrically about either side of the antenna origin. The UWB tag is placed in a radio-accessible location on the robot along its X-axis and affixed using 3M Command Velcro. First, one of the two points is selected in the Tracker software and the origin is dragged until it is snapped coincident with the point. Next, the frame is rotated about the X and Z axes with the adjacent point selected on the UWB tag to snap the Y axis in line with the adjacent point. Finally, both adjacent UWB tag spheres are selected in the Tracker software and the frame is dragged along the Y-axis until it snaps midway between the two. The only requirement for single point tracking is to align the UWB tag and Vicon object origins. The procedure just described still has one degree of freedom not constrained, the frame orientation about the Y-axis, which could be fixed by first utilizing three spheres known to be in the same plane.*





Figure 4.7: Clearpath Jackal with dual TREK1000 UWB tags installed and MoCap tracking points. The footprint of the Jackal is 17" x 20" [121]. Infrared light from a Velodyne lidar can be observed as captured by the cellphone camera.

Now, we are ready to collect data. Figure 4.8 below shows a single Ubuntu Linux machine used to collect Vicon data and UWB anchor data simultaneously. The Jackal has its own computer running an instance of ROS, but this is used only to teleoperate the vehicle in the MoCap arena.



Figure 4.8: Ubuntu laptop hooked up to UWB anchor0 and the Vicon system for data capture.

## Data Analysis and Discussion

The Jackal robot was driven around the entire MoCap arena in the ECR, and data was captured from the Vicon system and both dual UWB tags shown in Figure 4.7. First, we direct our attention to Figure 4.9, which shows the position of the forward UWB tag plotted against ground truth data, along with statistics on the UWB position error as measured using the Euclidean metric. Measured error is on the order of 10cm, the stated accuracy of Decawave's UWB system, although not at the 99% confidence interval that they claim in [54], which most likely refers to point-to-point ranging. Referring to Figure 4.11 for a histogram of the error data, it does appear to be somewhat normally distributed, although the Euclidean metric is constrained to be positive, so the distribution cannot be symmetric, and normalcy would instead be seen directly in each axis of sensor error. The distribution has a long tail, but the vast majority of UWB localization data points are within 30cm of their true position, agreeable to the later 95% reliability figure as cited in [54]. Overall, mean and standard deviation are not bad measures of the Euclidean error distribution. In the next chapter, we will look closer at the statistical distributions of individual coordinate components for use in a Kalman filter.

Turning our attention to the orientation data collected, let us look at Figure 4.10, which shows the heading of the Jackal robot calculated by finding the angle of the line created by the two UWB points, relative to their position on the robot. Although the general direction of travel remains consistent, the angle oscillates a great deal even when traveling in a straight line. This was predicted from the analysis of equation (3.4). The next chapter will revisit the concept of calculating heading angle from UWB tags with a larger robot capable of greater separation distance between the tags. The ECR lab space used in this chapter was simply too small to accommodate a robot larger than the Jackal. Nor did this limited space push the limits of UWB to see what it can do at range. Since the ECR's limits were exhausted with good results on UWB positional accuracy, the experiments were rerun outside and in a larger area to further characterize the Decawave TREK1000 UWB system. Those results will be discussed in the next chapter.

Comparison of VICON and Decawave TREK1000 UWB  
Trilateration using Decawave TWR algorithm: 110kbps, Channel 2

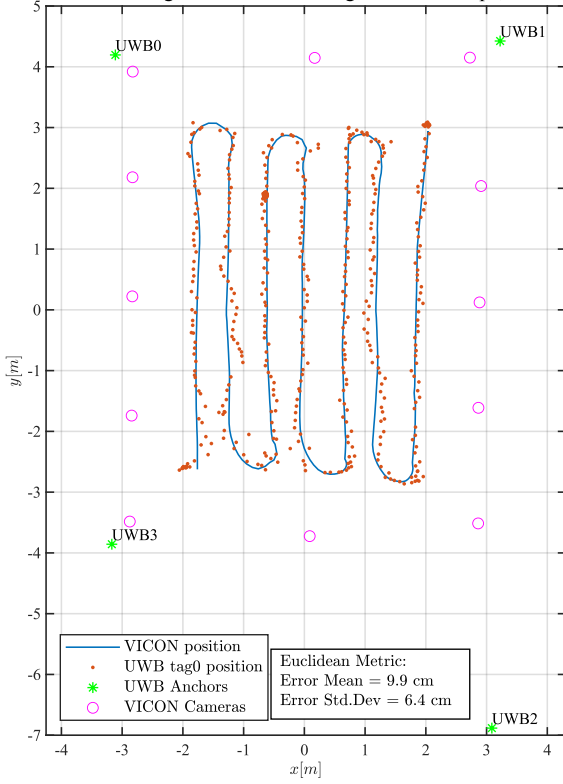


Figure 4.9: Point Data collected in the ECR from single-tag UWB localization of the Jackal robot.

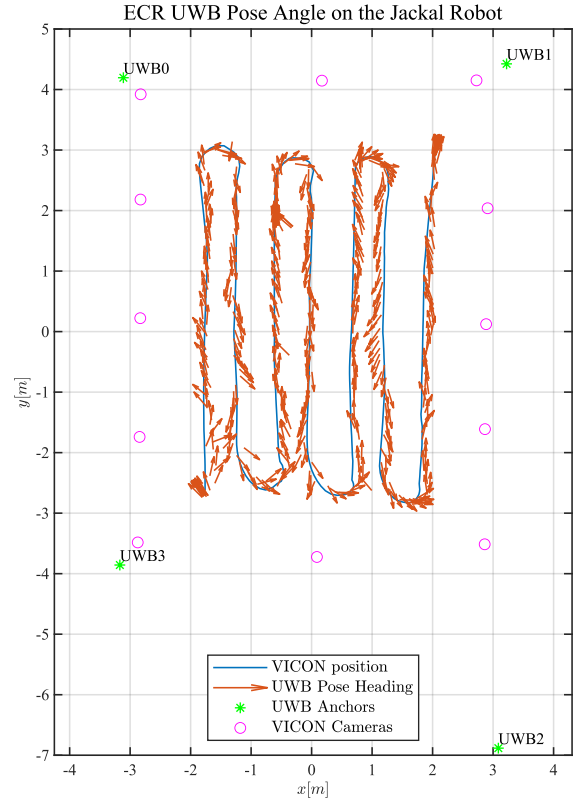


Figure 4.10: Orientation data collected in the ECR from dual-tag UWB localization of the Jackal robot.

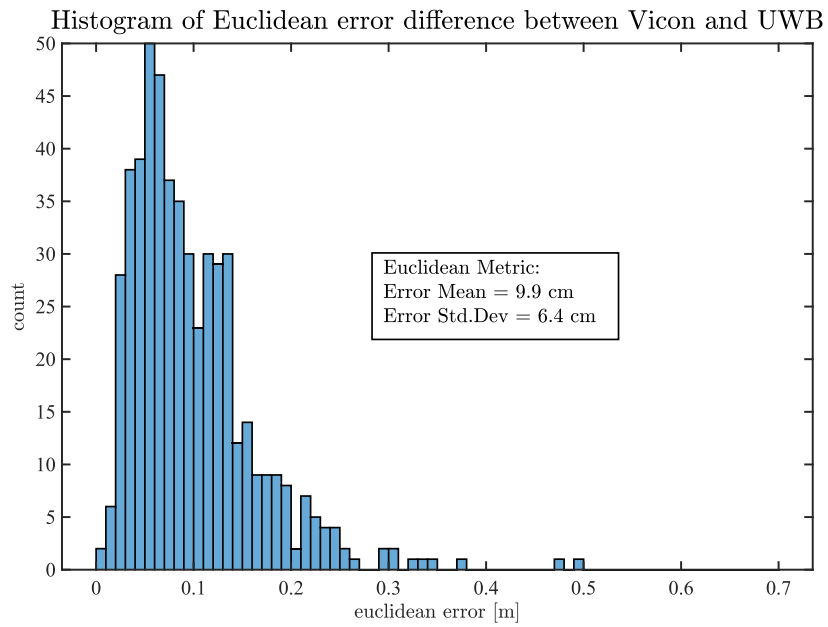


Figure 4.11: Calculated Euclidean error positional data for the single-tag UWB localization experiment performed in the ECR.

## CHAPTER 5: Large-Scale Decawave TREK1000 UWB Testing

This chapter includes further UWB ToA experiments conducted in larger and less structured environments than where typical UWB studies have been conducted in prior literature. First, an experiment is conducted outside to test the ultimate range of the UWB system. Then, we move inside a new indoor structure to statistically characterize UWB in a larger space than the ECR in the previous chapter.

### UWB Outside in a Field

The purpose of this test was simple: to test the ultimate range of TREK1000. There are many numbers published regarding the range of UWB systems. One oft-cited number is a range of 250-300m for Decawave UWB systems [54] However, the 250m range is for line-of-sight (LOS) conditions and the 300m figure refers to the absolute maximum ToF based on the time length of the window that the Decawave DW1000 chip accepts signals for. Most likely, both figures refer to point-to-point communication ranges. However, localization via trilateration requires at least three UWB anchors, which practically makes the point-to-point figures the maximum diameter of a circular tracking space as an UWB tag may be diametrically positioned from an anchor at this maximum range, even if another anchor is right next to the tag. This experiment was a practical range test of the UWB system.



*Figure 5.1: Some views of the outdoor UWB setup. The photo on the left shows the total station used to initialize UWB anchor positions along with the microhard radio station used to relay commands to the Jackal robot. The photo on the right shows the field used for testing with an UWB anchor on a tripod in the foreground and the Jackal robot in the distance.*



The experimental setup is shown in Figure 5.1 whereby the UWB anchors are setup as battery-powered on tripods in an outdoor field as shown in Figure 5.2. The Trimble S3 total station was once again used to initialize the UWB anchor locations, the coordinate results shown in Figure 5.2. The system was setup with a maximum range between anchors of approximately 70 meters, or about a quarter of the stated maximum range of the system. There was more space available, but this seemed to be a practical distance at which the tags stopped reliably and continuously communicating with the anchors, suggesting that 70m diameter is near the maximum LOS range of the TREK1000 UWB localization system.

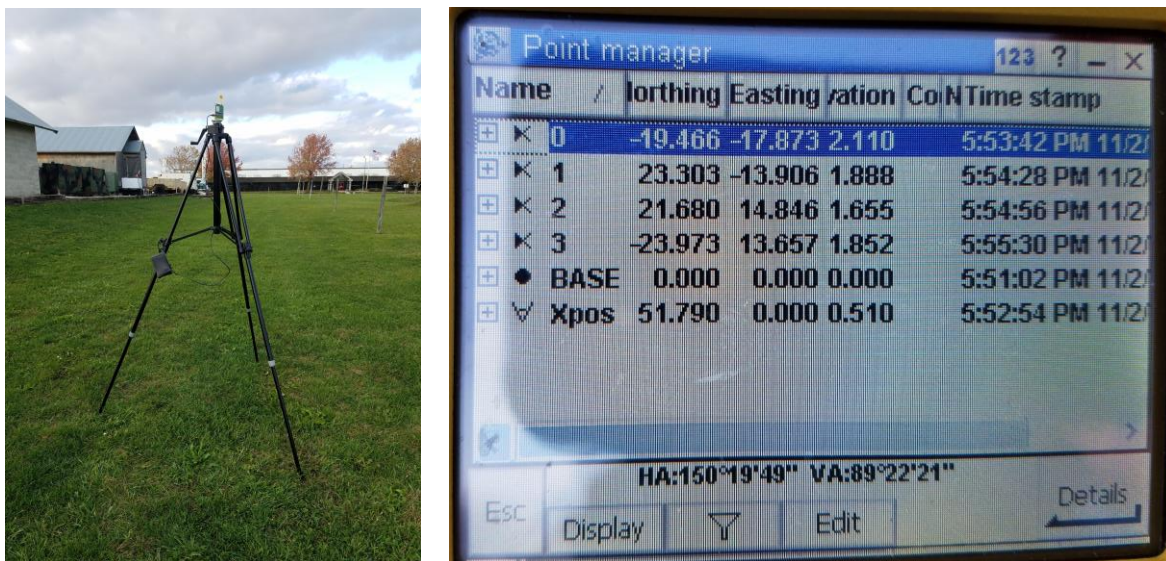


Figure 5.2: Outdoor range test UWB anchor mounting system and setup coordinates. The UWB anchors were powered by mobile battery packs, as can be seen hanging from the tripod on the left.

Localization results from the experiment are shown in Figure 5.3 and Figure 5.4. The path driven by the jackal robot is accurate as shown and could have been planned better for clarity of data. Figure 5.3 shows UWB point data from each of the dual tags mounted on the Jackal. Interestingly, there are many segments in which only points from tag0 or tag2 were localized. This suggests that the trilateration algorithm in the ROS driver was failing due to an insufficient number of tag to anchor ranges. The fact that either one tag or the other was still working suggests that the addressed tag frames were colliding with each other as described briefly in the previous chapter. This issue may perhaps be alleviated by upgrading the TREK1000 driver software from the factory shipped version, or by addressing the dual tags as 0 and 4. Because there are eight possible addresses for anchors and tags, 0-7, the selection of 0 and 4 would put

the maximum time interval between two tags in the data frame. This is something to be considered if one continues utilizing the TREK1000 system for robot localization. Otherwise, a robust time division system is recommended to avoid issues with conflicting airtime.

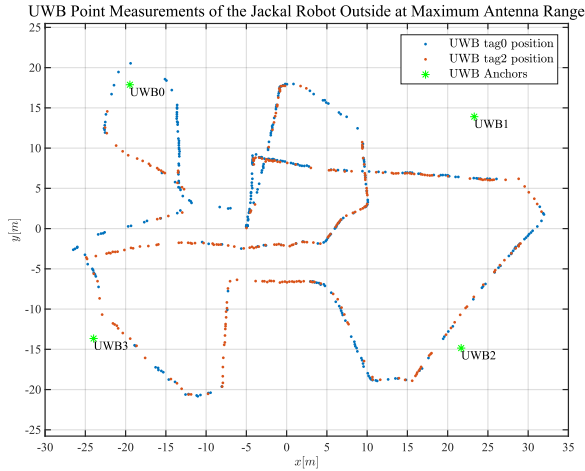


Figure 5.3: MATLAB plot of point position data from two UWB tags in the outside maximum ranging experiment using the Clearpath Jackal.

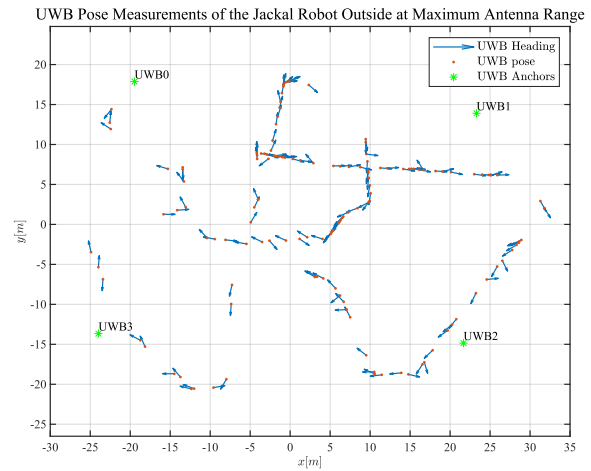


Figure 5.4: MATLAB plot of pose data from two UWB tags in the outside maximum ranging experiment using the Clearpath Jackal.

The results of pose estimation using dual UWB tags are shown in Figure 5.4. Due to the downstream nature of the ROS driver orientation calculation, failure to localize either of the UWB tags at any point also yields failure in calculation of orientation. Qualitatively, heading estimation is near the same accuracy as in the lab space, which is to say not the most accurate. The conclusion from this experiment is that the practical range for a Decawave TREK1000 system is a circle of at least diameter 70m for single-tag point tracking and most likely dual-tag orientation tracking as well, given the correct system settings.

### UWB Inside the Tent

Figure 5.5 shows several views from outside of the temporary structure in which the rest of the data was taken for this chapter. It will be referred to as the tent from this point forward. Shown in Figure 5.6 are inside perspectives of the tent including the two modular office spaces inside, one two stories tall, and the Vicon MoCap system setup in a larger area on taller tripods to allow for tracking of larger robots.





Figure 5.5: Views of the tent. A full-sized piece of construction equipment and the man doors on the structure give some scale to the large size of this facility.

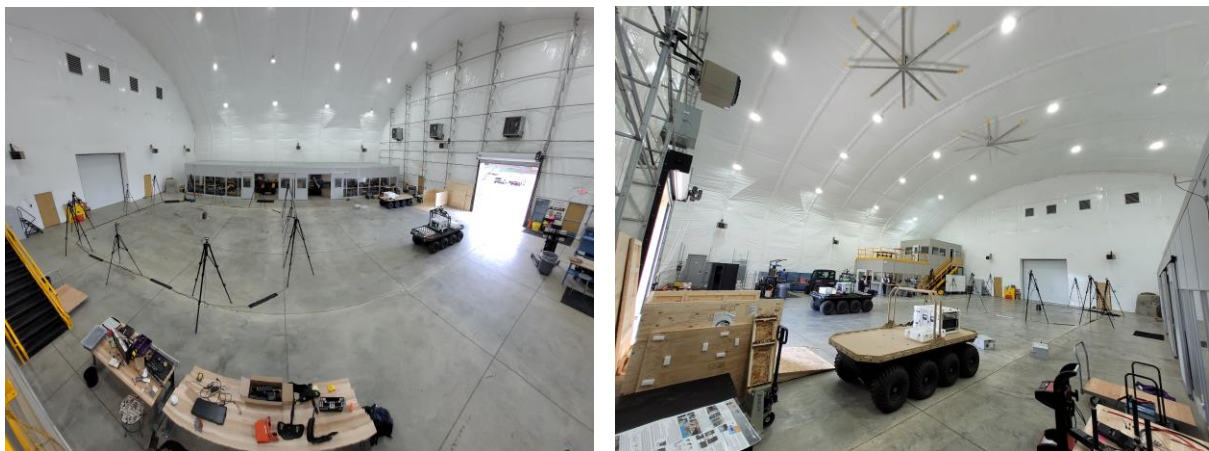


Figure 5.6: Inside of tent showing the Vicon arena covering not quite half of the available inside floor space along with the two modular office structures and two J8 robots.

The Vicon Visualization Tool currently available on Vicon's website predicted operating accuracies are shown in Figure 5.7 for the tent Vicon tracking volume setup. Generally, tracking

has been good, however quarter-inch passive markers were used for the following experiments and this required fully opened apertures on the cameras. It may be beneficial to use larger passive markers for better centroid definition in the future.

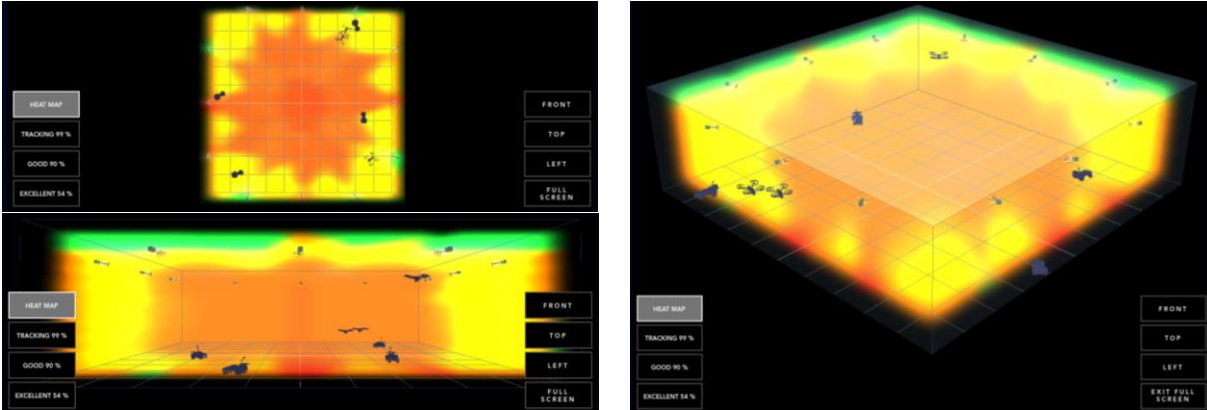


Figure 5.7: Vicon Visualization Tool results for a robotics application system with 12 Vero v2.2 cameras in a room of size 10 x 10 x 3m tracking volume. The tool shows a tracking volume of 99%, good of 90%, and excellent of 54% for the camera configuration used in the tent.

A larger space demands a larger robot, and that relative fit was achieved using the Argo J8 XTR robot, as shown in Figure 5.8. This robot is the size and weight of a small car and can be teleoperated using LOS RF. It has a large platform for mounting various sensors and payloads.



Figure 5.8: Argo J8 XTR robotic platforms in multiple colors. Base weight of 1650lb and a footprint of 61" x 116" [128].

For completeness, the tent UWB anchor setup coordinates are shown in Figure 5.9 as measured by the Trimble S3, with the same transformation to cartesian coordinates as discussed earlier.

Name	Northing	Easting	Elevation	Time stamp
0	-5.971	-9.522	3.848	4:55:19 PM 8/27/17
1	9.474	-9.213	2.990	4:56:14 PM 8/27/17
2	9.446	10.187	2.976	4:56:49 PM 8/27/17
3	-7.176	13.766	2.893	4:57:22 PM 8/27/17
BASE	0.000	0.000	0.000	4:47:01 PM 8/27/17
Xpos	9.557	0.000	0.940	4:53:03 PM 8/27/17

HA:117°31'55" VA:84°58'53"

Figure 5.9: Tent UWB anchor setup. Northing = X+. Easting = Y-. Elevation = Z+. Note that anchors were decremented such that anchor0 became anchor3, anchor1 → anchor0, and so on for this chapter, but not the next chapter.

The method of mounting and measuring location for the J8 UWB tags is shown in Figure 5.10. The Vicon object was setup with its origin centered on the marker at the center of the vehicle's rotation. Tape measures were then used to find UWB tag offsets relative to this central point for input into the launch file. The Vicon Tracker object's X-axis was pointed toward the marker forward of center and the Z-axis was pointed upwards, using markers perpendicular to the vehicle's X-axis to snap rotation, to align tag with the vehicle frame.

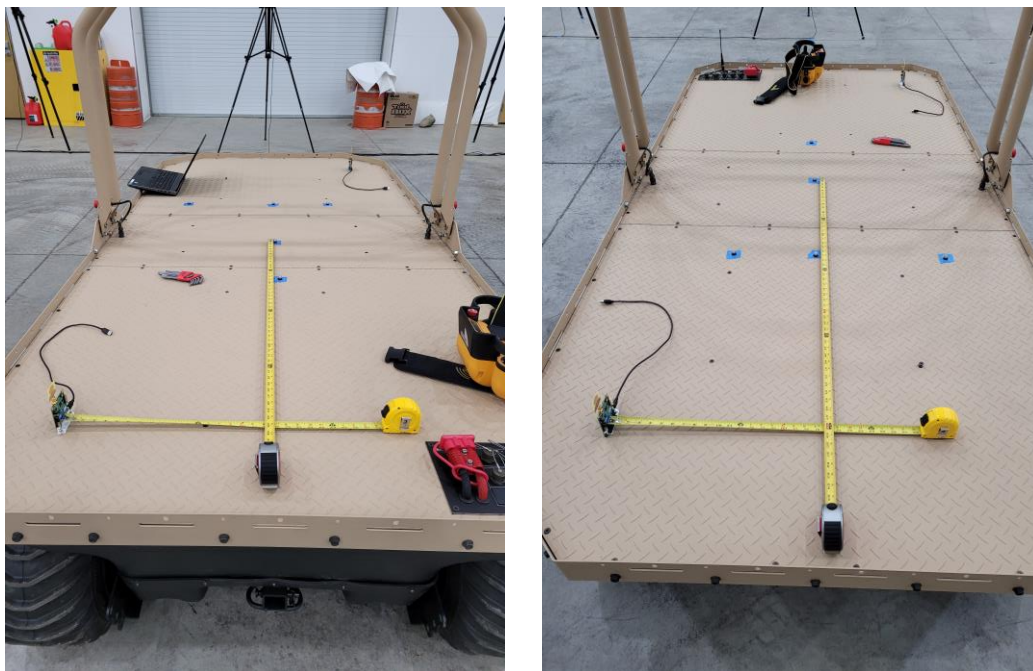
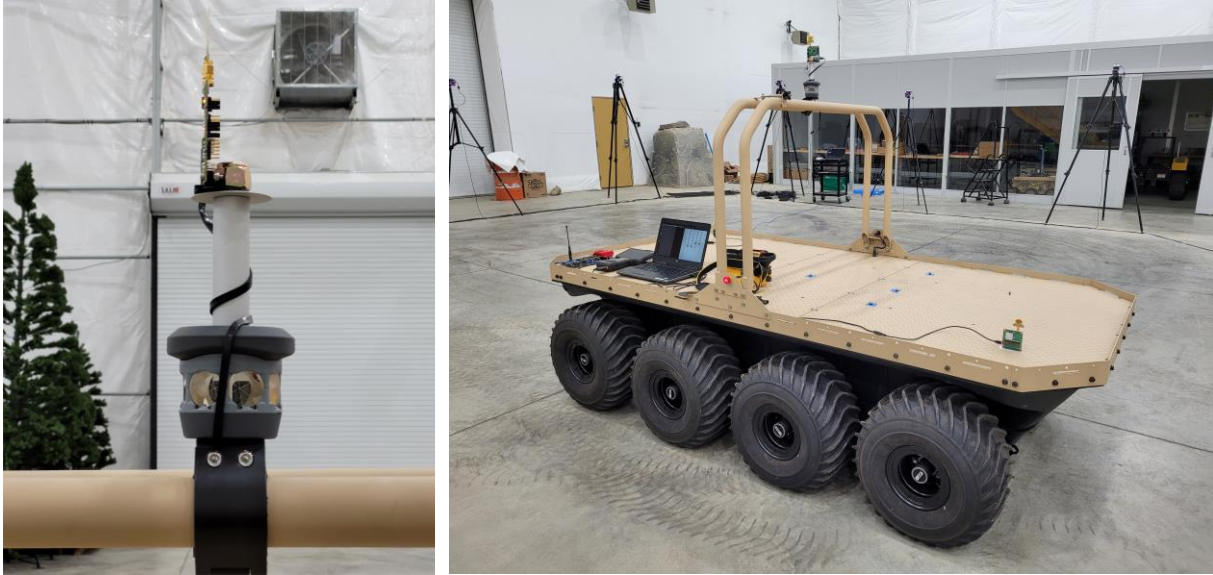


Figure 5.10: Measuring UWB tag locations from the Vicon center. Leading UWB tag0 location WRT to measured MoCap marker is (1.23825, -0.47879, 0.0). Trailing tag2 location WRT the same point is (-1.152525, 0.48641, 0.0).



Another UWB tag was mounted directly above the center of the vehicle, as shown in Figure 5.11. This was used to collect UWB single-tag point data for comparison with pose data from the UWB dual-tag setup.



*Figure 5.11: On the left is mounted a single UWB tag directly over the center of vehicle rotation and MoCap marker where the vehicle object's origin was centered. On the right is shown the J8 fully outfitted with 3 UWB tags and a laptop to power them and collect data.*

### Frame Alignment using Horn's Method for Absolute Orientation

The same amount of effort for aligning the Vicon and UWB frames did not occur in the tent as it did in the ECR. That level of control did not seem productive for the energy expended and is not a realistic expectation for systems outside of a lab environment. Instead, the MATLAB package ABSOR [129] was used in post-processing to calculate the coordinate transform required to align frames via translation and rotation. This package implements Horn's method for finding the absolute orientation using unit quaternions [130]. It contains a single function, `absor()`, that takes as input two sets of 2D or 3D coordinates of the same length that are assumed to be point clouds of the same data, collected from different locations. It returns a rotation matrix and optional translation vector and scaling factor that minimizes the least squares error between the two sets of data. Scaling factor was not used here under the assumption that each absolute localization sensor measures to the same real-world scale.

Horn's method was developed for photogrammetry tasks, but the same problem is present here, as the localization data being collected are measurements of the same state estimate from systems with independent coordinate frames. It should be noted that minimizing the relative frame error between localization sensors is, in general, a desirable activity in robotics. This method could be used online (via another implementation) or offline to provide static transformations between sensor systems to ROS. For example, if MoCap was only available in a limited region, its origin and axes could be used as the map frame and all other sensors statically transformed to that frame. The calibration process would be to start the robot, not move for a time while collecting static data and then drive in a straight line to collect a dynamic point cloud. ROS approximate time filters would run to create directly corresponding point clouds for use in Horn's algorithm. Once calibrated, the robot could leave the MoCap arena and drive wherever desired.

## Results

The results of data collection from separate runs using single-tag and dual-tag UWB are shown graphed against ground truth MoCap data in Figure 5.12 and Figure 5.13, respectively. The Euclidean metric error histograms for each setup are plotted in Figure 5.14 and Figure 5.15, respectively. Very similar error results were obtained for each method, but the dual tags were slightly better in accuracy. This may be attributed to a scenario where a single tag may give outlying data, but a second tag is mitigating it with better data. The dual-tag method of robot position calculation is to average the results of each tag together. Conceptually, this also corresponds to the situation with Kalman filtering in which the combination of multiple noisy sensors results in a sensor fusion measurement that has smaller variance than any of the individual sensors. Overall, both runs have larger error by 5-10cm more than in the previously discussed ECR experiments in both mean and standard deviation, but this is expected in a larger space. Intuitively, one would imagine that ToF distance measurements have a fixed error proportional to the magnitude of their ToF measurements and that this property propagates into trilateration calculations. Overall, UWB accuracy is better than GNSS would be if it were even available inside. The measured UWB accuracy is two orders of magnitude less accurate than the MoCap system, but at a cost reduction also of two orders of magnitude.

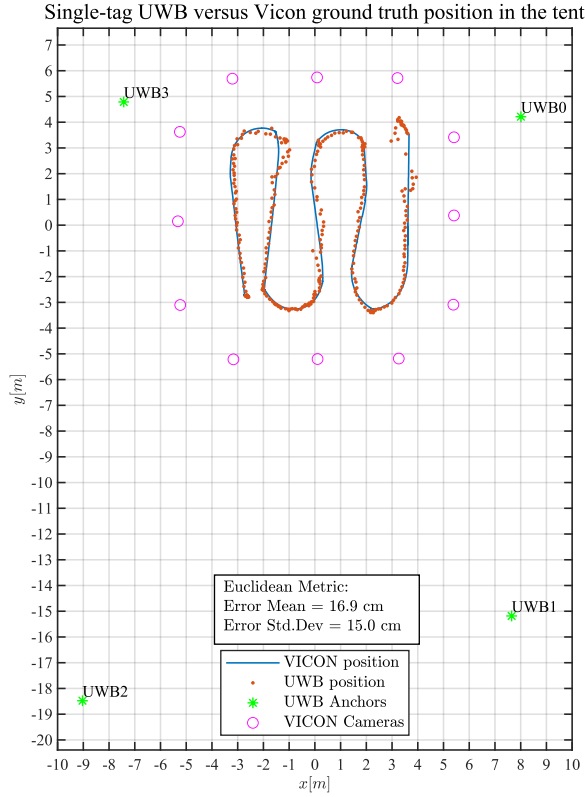


Figure 5.12: Comparison of ground truth Vicon MoCap data vs single-tag UWB point data captured in the tent.

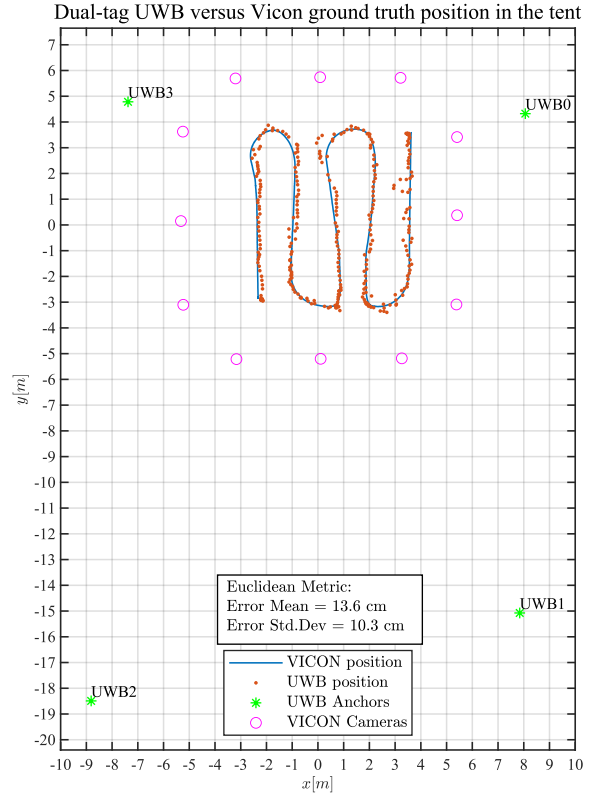


Figure 5.13: Comparison of ground truth Vicon MoCap data vs dual-tag UWB average point data captured in the tent.

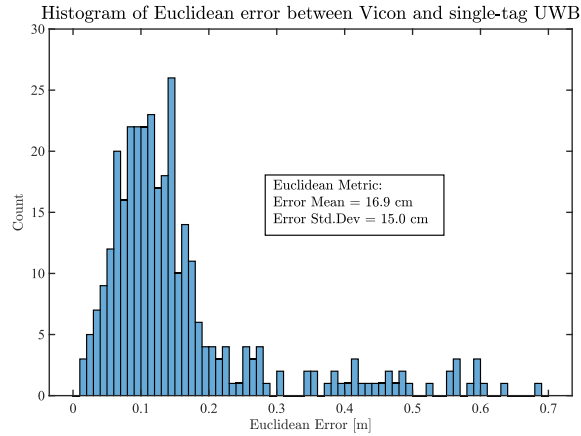


Figure 5.14: Euclidean error comparison of ground truth Vicon MoCap data vs single-tag UWB point data captured in the tent.

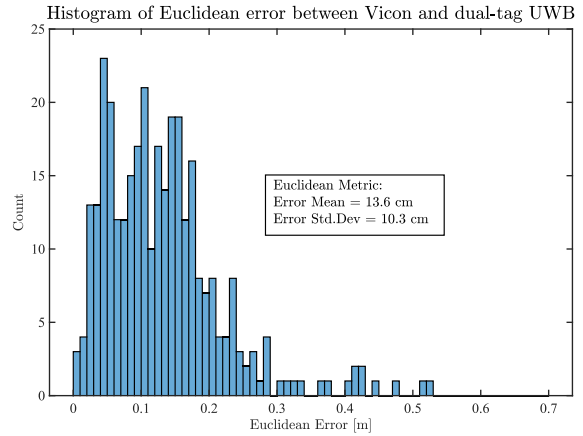


Figure 5.15: Euclidean error comparison of ground truth Vicon MoCap data vs dual-tag UWB average point data captured in the tent.

Turning our attention to measurements of heading, Figure 5.16 shows the position and angular heading as recorded by the Vicon system, while Figure 5.17 shows the heading as recorded by the UWB system overlaid on the Vicon ground truth positions for ease of visualization. We can

see that the heading graphs are overall quite consistent. Driving in a straight line does not see the same angular oscillations as when then two UWB tags were only one foot apart on the Jackal in the ECR. Now, the tags are over four feet apart and angle readings look stable.

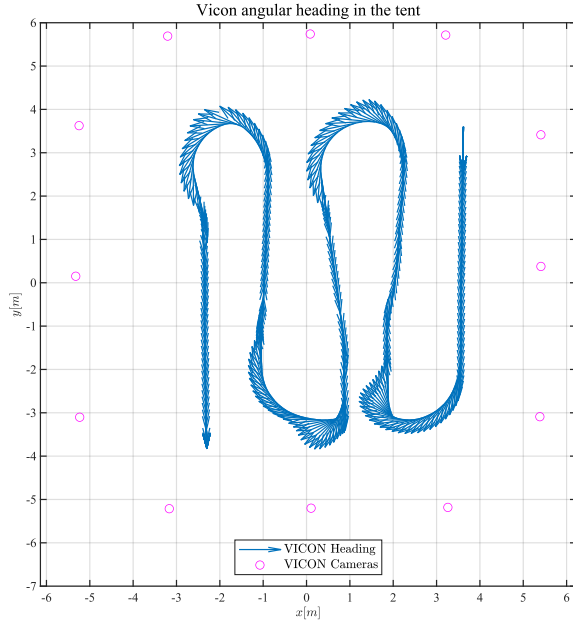


Figure 5.16: Heading data captured by Vicon MoCap of the J8 robot's heading in the tent.

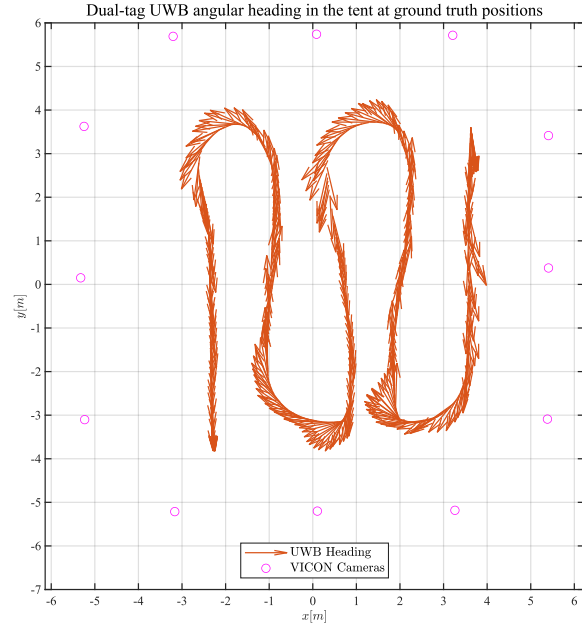


Figure 5.17: Heading data at ground truth positions captured by dual-tag UWB of the J8 robot's heading in the tent.

Angular heading stability can be quantified by calculating the point-to-point difference in heading as recorded by UWB versus ground truth. The histogram and statistics of this calculation are shown in Figure 5.18. The data appears to be symmetrically and normally distributed with an almost zero mean of 0.4 degrees. The standard deviation is 5.8 degrees, but this looks to be driven by outliers as most of the data is bounded by  $\pm 15$  degrees difference error. Like all UWB sensor values, this noisy data can be smoothed by using an appropriate filter such as a Kalman filter or particle filter. Indeed, a Kalman filter requires a matrix of sensor covariances and these statistics can be directly read from the data calculated here. Figure 5.19 shows a histogram of UWB error in the X-coordinate and Figure 5.20 shows the same for the Y-coordinate. Both distributions are also symmetric and normally distributed with zero-mean. Their standard deviations are comparable to the Euclidean error and look to both contribute equally. These values should be converted to the appropriate unit, squared, and used directly as diagonal variance values in the Kalman filter sensor covariance matrix.

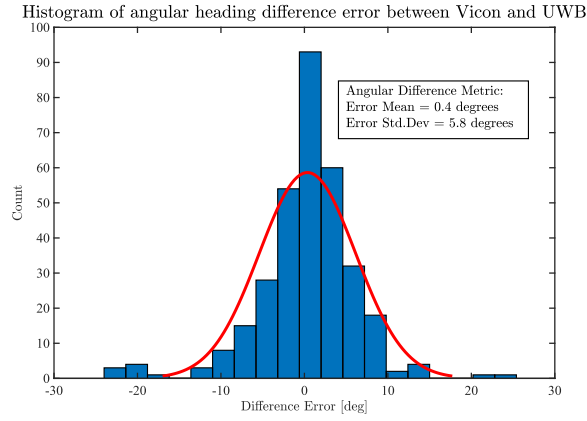


Figure 5.18: Difference error comparison between ground truth Vicon MoCap data vs dual-tag UWB average point data captured in the tent. Graph shows angular difference in heading of UWB vs Vicon. Normal distribution fit is shown in red.

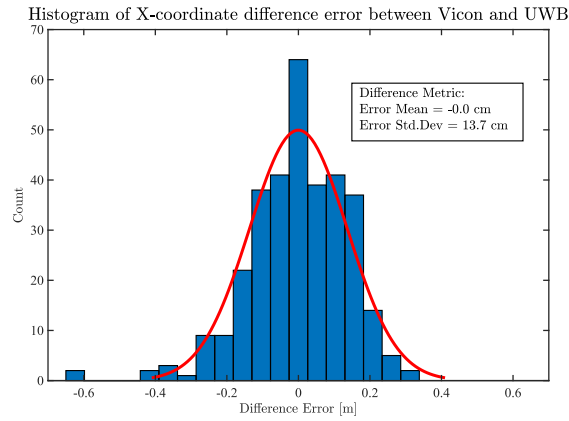


Figure 5.19: Difference error comparison between ground truth Vicon MoCap data vs dual-tag UWB average point data captured in the tent. Graph shows difference in X-coordinate of UWB vs Vicon. Normal distribution fit is shown in red.

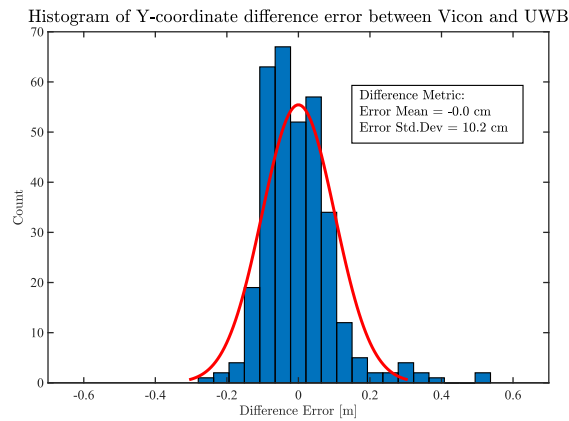


Figure 5.20: Difference error comparison between ground truth Vicon MoCap data vs dual-tag UWB average point data captured in the tent. Graph shows difference in Y-coordinate of UWB vs Vicon. Normal distribution fit is shown in red.



Figure 5.21 below shows the positional and angular errors between the dual-tag UWB sensor data and Vicon ground truth data. Note that larger positional errors lead to larger angular errors, as is to be expected from the method of orientation calculation.

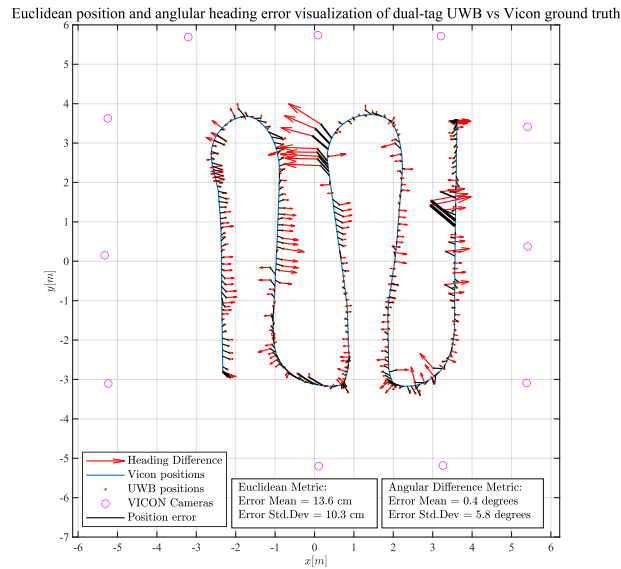


Figure 5.21: Graphical representation of the positional and angular error magnitudes of dual-tag UWB sensor data vs Vicon MoCap ground truth data for a J8 robot in the tent. For position, error magnitude is proportional to line thickness. For angle, error magnitude is proportional to arrow length. Angle error is a function of position error due to the method of orientation calculation from positional data.

## Discussion

This chapter has demonstrated the Decawave TREK1000 at full-scale capabilities and statistically characterized the UWB sensor for robot localization. Standard deviation data has been given at enough precision in the plots to accurately populate the Kalman sensor covariance matrix. However, a Kalman filter also has the process covariance matrix to capture process noise. This thesis assumes that all noise is from the sensor itself and not a result of the environment, but this is not entirely true. The Argo J8, for example, is relatively tall and has low pressure tires with mud ridges. It may sway and jostle, which is a process variant that is being captured here as sensor noise. Additionally, UWB performance will degrade in cluttered environments with NLOS conditions. A possible way to define sensor and process noise is as intrinsic and extrinsic to the system, respectively. Then, for each environment, process noise will have to be characterized. This is outside of the scope of this thesis but is another avenue of research that could be pursued.

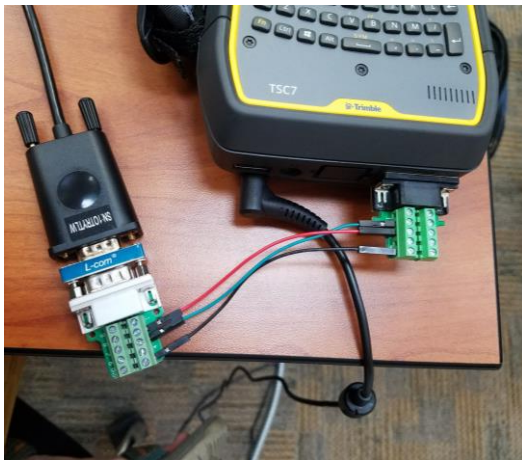
## CHAPTER 6: Trimble SX-10 Total Station Localization

This chapter introduces another absolute measurement sensor with high-accuracy positioning capabilities. While Vicon MoCap provides millimeter-level accuracy for full pose information and multiple object tracking, the system is large, static, and takes a great deal of time to setup. Its best use-case is in a fixed environment where it can stay installed or with a large team of trained people to set it up and break it down when needed. The MoCap system range is limited by the number and model of cameras used and ultimately by budget.

The Trimble SX-10 Scanning Total Station has all the functionality of the Trimble S3, plus many other features. The ability that sells this newer model is that of combined laser and camera scanning to create RGB point clouds of an environment, as well as stitch multiple of these scans together to enable full 3D representations of landmarks and areas of interest [131]. These benefits make the SX-10 an interesting instrument for creating calibration data. It may be possible to compare a lidar scan of a controlled environment to an SX-10 scan using the iterative closest point (ICP) algorithm or by using a technique where a mesh is created and evenly sampled for each of the point clouds to reach a condition where it is possible to utilize Horn's method to calibrate sensor installation locations.

The feature that makes the SX-10 interesting for this thesis is its ability to track a moving target while outputting serial data. This is possible by changing the measurement setting to tracking, locking on to a retroreflective prism, and opening the serial data menu to begin streaming data. A simple ROS driver was written to parse the SX-10 serial data, transform left-handed spherical coordinates to cartesian coordinates, and publish the resulting data as a PointStamped message. Figure 6.1 shows the null modem serial connection that is required to connect the between Trimble TSC7 controller and serial to USB converter cable for computer connection. This is not documented in the SX-10 literature. The TSC7 connects to the SX-10 wirelessly at long-range and can be placed either on the robot or another convenient location off the robot. The SX-10 must be setup on a tripod that has LOS to the prism. A 360-degree prism, Figure 6.2, was used in these experiments to allow tracking of the J8 at any heading orientation. The SX-10 returns left-handed spherical coordinates in (HA, VA, EDM) format, but horizontal angle zero is

still the X-axis after conversion to cartesian coordinates. The SX-10 sets HA to zero on startup and will retain this information unless changed by the user. For this reason, it is best practice to point the SX-10 towards the map positive X-axis on startup to aide in system sanity checks. The vertical angle is absolute, relative to gravity, and the system level must be fine-tuned first by the bubble level on setup and then by the electronic level on startup.



*Figure 6.1: Null modem required for Trimble SX-10 serial output. An inexpensive null modem unit can be found online.*



*Figure 6.2: 360 Prism for tracking from any heading. The prism is mounted upside-down here using double-sided tape.*

## Results

The setup for a long-range outdoor experiment is shown in Figure 6.3. The parking lots and roads directly outside of the tent were used to drive the around J8 on and record position data with the SX-10. Figure 6.3 is a panorama where scale is hard to determine, but the plotted data in Figure 6.4 is overlaid on satellite imagery that is easier to comprehend the road layout with.



*Figure 6.3: Outdoor area for collecting Trimble SX-10 data. The two roads in the right half of the picture are perpendicular.*

The data shows good qualitative correspondence with the actual path driven. Most impressive is the ability to track at ranges over 200m. Of course, as an optical instrument, LOS is always required. The SX-10 datasheet claims Autolock tracking of a 360 prism at a range of up to 300m in normal conditions and 700m in perfect conditions [132]. The data shows two interesting phenomena. One is the discrete grouping of several points of data and the other is the lateral jumping that seems to have occurred downrange. Nothing could be found in the driver code to cause these issues.

It is believed that the point groupings are output from a filter internal to the SX-10, such as a Kalman filter, another process taking up computing time, or some other cause of periodic delay in serial output. The lateral jumps were hypothesized to be either encoder discretization error or floating-point error of 32-bit floats but after reviewing the data sheet and output, both were out. The current hypothesis is that the SX-10 laser switching between two retroreflector windows on the 360 prism is the cause of the jumping. This could occur if the prism is even just slightly perturbed from perfect calibration.



Figure 6.4: Outdoor Trimble SX-10 data. The satellite imagery is outdated. There were no cars present at the time of data collection, the tent is not shown, and the grass section in the bottom left of the image has been scaled back for more pavement. This data shows characteristic point groupings of data along with discrete jumps in the SX-10's lateral direction.



To characterize the SX-10 accuracy quantitatively, an experiment was conducted inside the tent MoCap arena to calculate point-to-point statistics as with the UWB sensors in the previous chapters. Figure 6.5 shows the SX-10 setup within the tent, located on the second floor of a modular office structure to facilitate LOS tracking of the J8 above the Vicon camera tripods without taking any space in the tracking volume.



Figure 6.5: Trimble SX-10 “Eye-in-the-Sky” to track inside the tent and see above the tops of the Vicon cameras on tripods.

To align the origins of the Vicon and SX-10 frames on the robot in the 2D plane, a plum-bob was fashioned as shown in Figure 6.6 to place a MoCap marker directly below the 360 prism. As always, this is required to avoid compensating for a relative coordinate transformation between sensors mounted on the robot. If sensor origins are aligned, the data will show identical paths in space, whereas any origin offset will cause the two sensor paths to differ. Ideally, sensor mounting locations on the robot would be precisely known ahead of time, but that requires a great deal more mechanical constraint than is available on the stock J8.



Figure 6.6: Centering the Vicon origin  $(x, y)$  coordinates to the Trimble SX-10 total station  $(x, y)$  coordinates by improvising a plumb-bob for alignment by gravity. This treatment implicitly assumes that the floor to operate on is currently level.

If the two sensor paths are identical, but transformed from each other, then Horn's method for calculating absolute orientation can be used to find the transformation matrix, as discussed in the previous chapter. The results for the overlaid paths are shown in Figure 6.7, which indicates better sensor accuracy than was the case for UWB. The discrete point groupings can still be seen even at this shorter range. Taking a closer look at this distribution in Figure 6.8, most of the error falls between the 2-3cm range, with a rectangular distribution otherwise. It would be a stretch to call this error normally distributed.

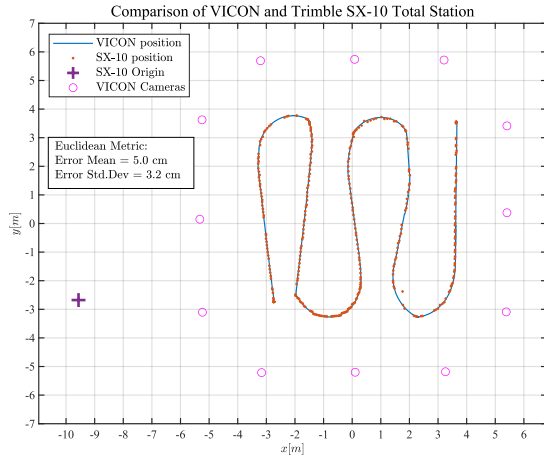


Figure 6.7: Comparison of sensor position data between the Vicon MoCap system and the Trimble SX-10 inside the tent.

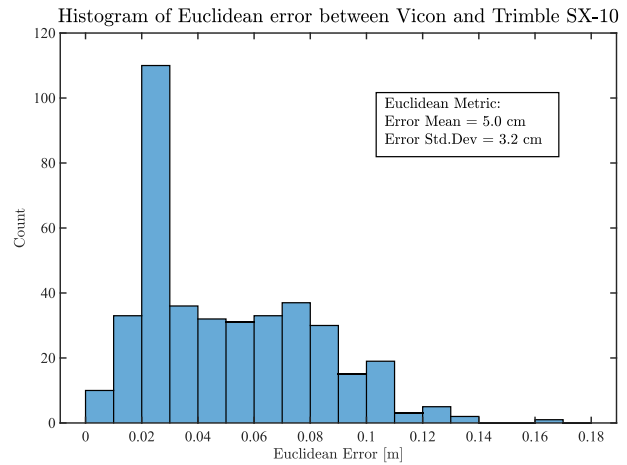


Figure 6.8: Euclidean error comparison of ground truth Vicon MoCap data vs Trimble SX-10 data captured in the tent.

To better investigate the Euclidean error distribution, we should examine each of the individual error components. Our analysis is in 2D, so we should look at the error distributions for X and Y. These are shown in Figure 6.9 and Figure 6.10, respectively. Note that the distributions look quite different from each other, and this is not unexpected, as each measurement comes primarily from a different sensor. Due to the placement of the SX-10 relative to the MoCap arena, X-coordinate measures are dominated by EDM and Y-coordinate measures depend mostly on the HA encoder. Perched on the second story, the VA encoder also plays a role in the calculation from spherical to cartesian coordinates, but we will ignore this axis. The EDM looks like it most frequently measures short, but only 5cm short as a rule, with an asymmetric tendency to measure long up to 10cm. The HA encoder error is symmetric with an error spike just on either side of zero, but a greater frequency of larger errors than the EDM. The statistics shown can be placed directly into a Kalman filter sensor covariance matrix, however this pattern would not hold if the drive path deviated from isolating these axes as independent through spherical to cartesian coordinate transformation.

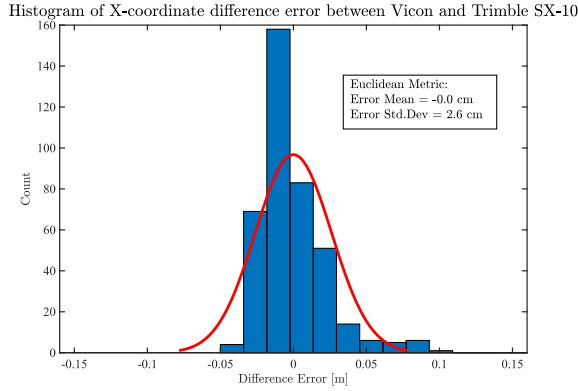


Figure 6.9: Histogram of X-coordinate difference error for Trimble SX-10 vs Vicon MoCap. X-axis measurements are dominated by the laser EDM of the total station. Normal distribution fit is shown in red.

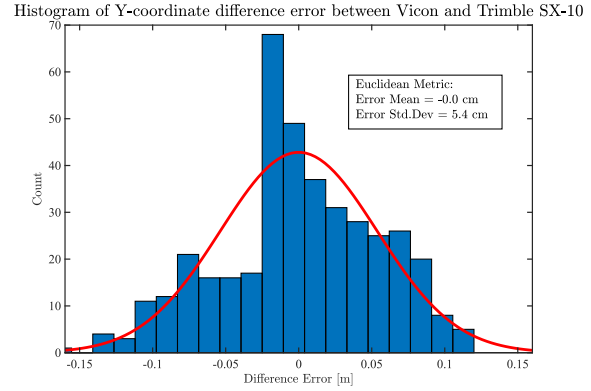


Figure 6.10: Histogram of Y-coordinate difference error for Trimble SX-10 vs Vicon MoCap. Y-axis measurements are dominated by the HA encoder of the total station. Normal distribution fit is shown in red.

A more proper statistical treatment of the Trimble SX-10 would calculate variances in native spherical coordinates and then transform these variances along with the data into a cartesian representation into properly converted uncertainties for the covariance matrix. Published axis accuracy can also be found in the datasheet [132]. As this thesis is focused on UWB and using the SX-10 as a tool, we will let the reader further investigate the SX-10 and return to discussion of UWB with the knowledge that the total station's accuracy is somewhere between that of the MoCap system and Vicon.

### UWB to SX-10 Comparison

The final test of the Decawave TREK1000 UWB in this thesis was to compare it against the Trimble SX-10 in a larger space than the Vicon MoCap could support. To this end, both systems were operated along the entire length and width of the tent available for robot testing. Both single-tag and dual-tag UWB methods were compared against the SX-10 and these are shown in Figure 6.11 and Figure 6.12, respectively. The results are similar to those with the MoCap system where dual-tag UWB error statistics are slightly lower than single-tag, but otherwise both methods track well. The SX-10 is contributing to overall error as well, but the magnitude of error from UWB is higher, as can be seen in the corresponding histograms of Euclidean error in Figure 6.13 and Figure 6.14 which look similar to the metrics from the MoCap system, albeit with perhaps slightly longer tails.



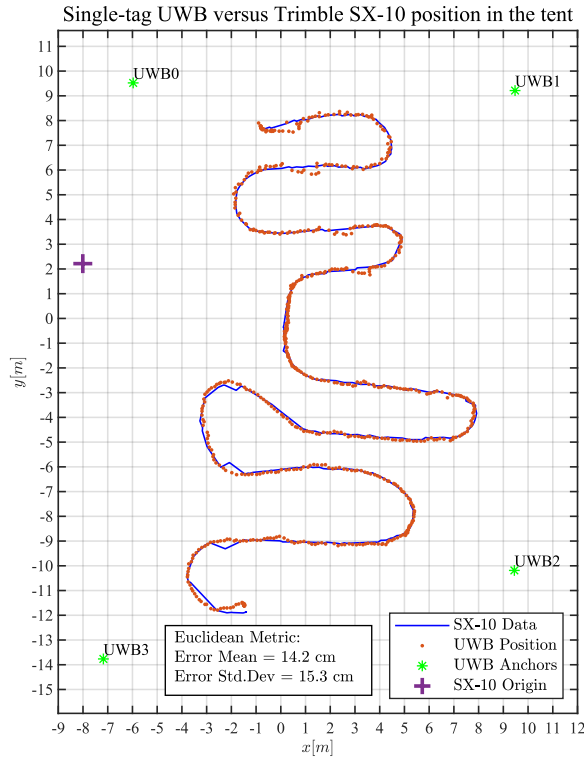


Figure 6.11: Comparison of sensor position data between the Trimble SX-10 and single-tag UWB inside the tent.

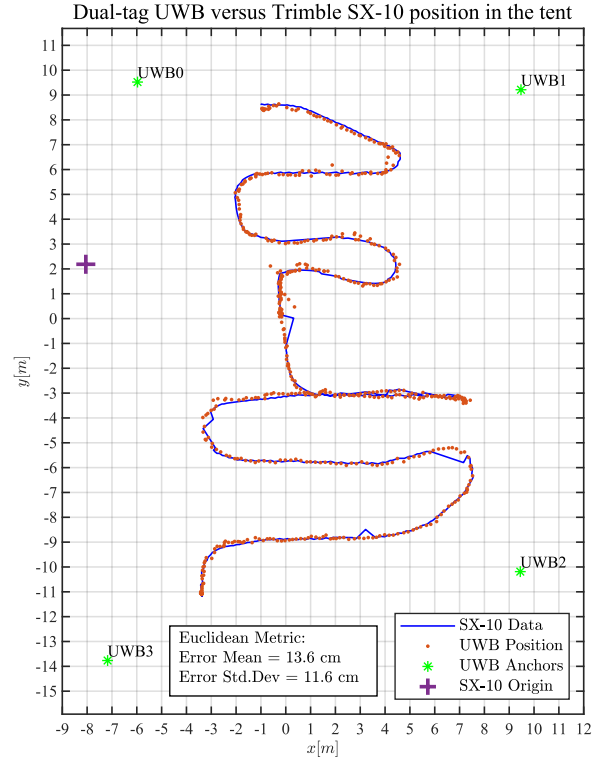


Figure 6.12: Comparison of sensor position data between the Trimble SX-10 and dual-tag UWB inside the tent

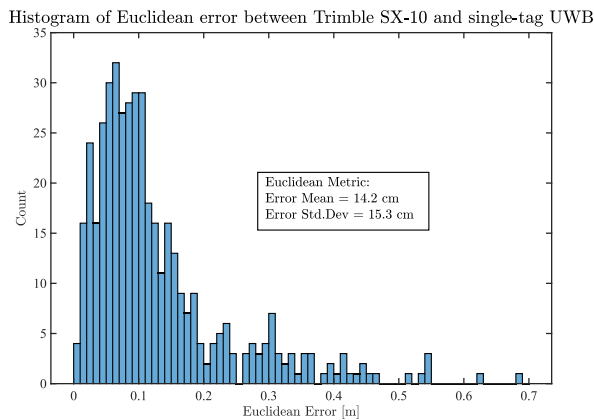


Figure 6.13: Comparison of Euclidean error metric between the Trimble SX-10 and single-tag UWB inside the tent.

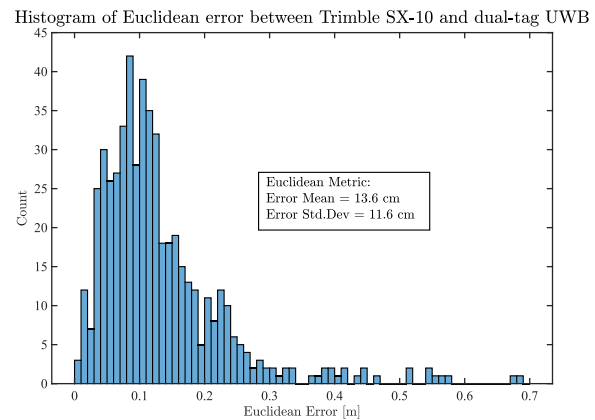


Figure 6.14: Comparison of Euclidean error metric between the Trimble SX-10 and dual-tag UWB inside the tent.

The final plot to look at for the Decawave TREK1000 UWB system is Figure 6.15, which shows full 2D pose data from dual-tag UWB versus position data from the SX-10. One of paths overlaps, but we can see good tracking in all three variables (X, Y, Heading) across the entire tent. Heading can only be evaluated qualitatively, but by that measure it looks quite good.

# Dual-tag UWB versus Trimble SX-10 position in the tent Heading is calculated using dual-tag UWB

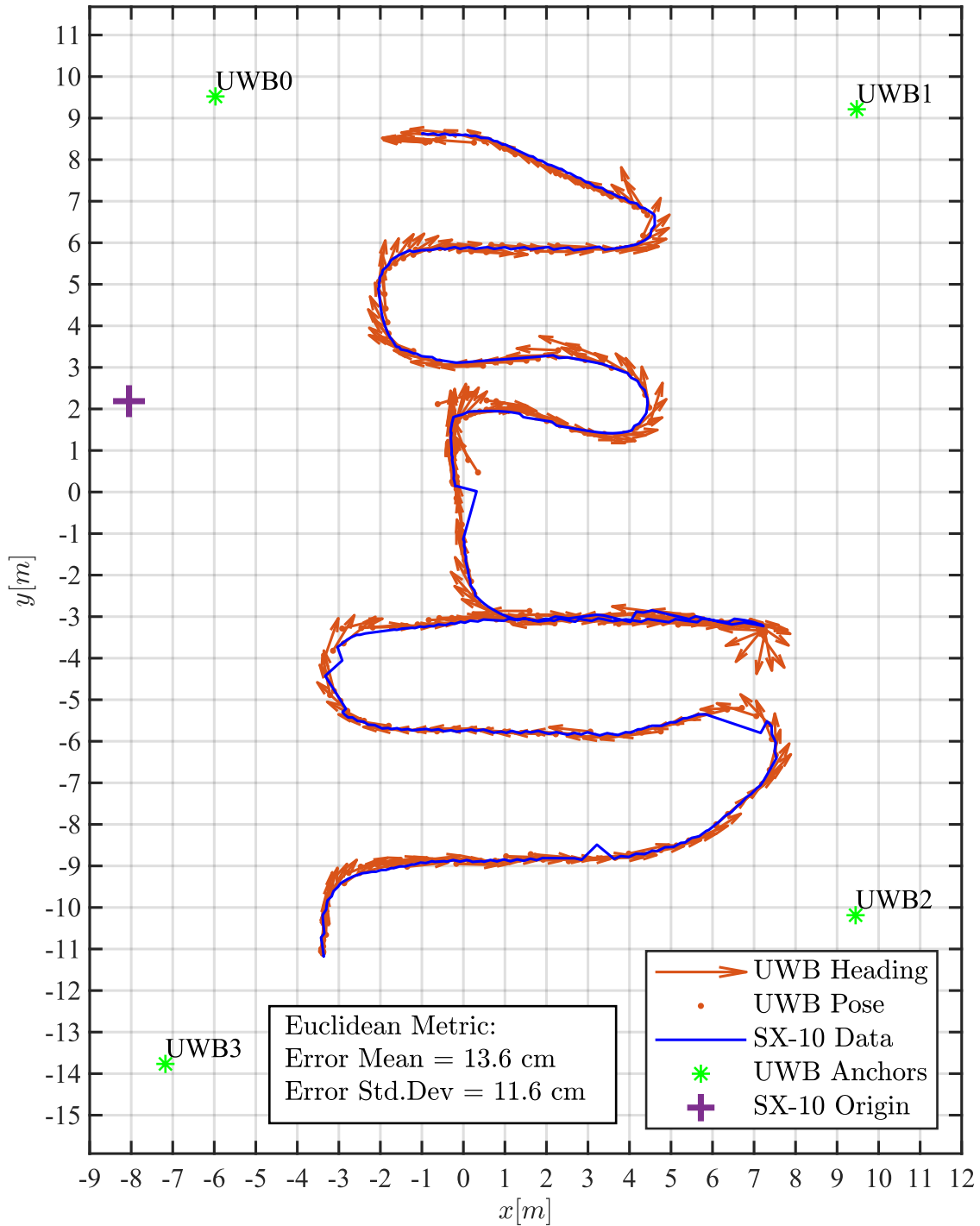


Figure 6.15: Quantitative comparison of robot position data from dual-tag UWB vs the Trimble SX-10 total station. Qualitative visualization of dual-tag UWB heading data. This plot verifies the Decawave TREK1000 UWB capability to operate within the entirety of the tent's robotic driving range fully and accurately.

## Discussion

This chapter has introduced the Trimble SX-10 Total Station as a stand-in absolute measurement sensor in place of motion capture systems at greater ranges to confirm UWB performance. The instrument need not only be used as verification; it could also be a powerful tool for robot localization. Although we briefly investigated the statistical measurement errors of the SX-10, there is more to investigate for the continuous tracking mode used here.

Despite the SX-10's promise as a stand-alone robot localization sensor, there is still further system optimization needed to achieve this status. The instrument was exploited here in a way that it was not designed or marketed for. If Trimble focused engineering on this mode of the system, it would be expected to work better, as currently the accuracy is much below that specified for surveying in the datasheet [132]. Areas of improvement include lowering latency, eliminating data output grouping, and increasing output data rate. Further, integrating the SX-10 with ROS could even include the hardware in a control loop to rapidly regain LOS if the prism view were to be obstructed. These improvements could make the SX-10 a highly desired absolute sensor for robot localization in real-world scenarios, especially in GNSS-denied environments. Indeed, preliminary accuracy studies seem to indicate that the system is much more accurate than standard GNSS and possibly on par with highly accurate differential-GPS systems. Further study is recommended to quantify the system accuracy at long-ranges and in a variety of operating conditions, such as comparing the system outside against a highly accurate KVH GEO-FOG IMU [133] to be used as ground truth data.

## CHAPTER 7: Single-Anchor UWB Localization using Multiple Differential PDoA Antenna Pairs

While UWB ToA is an excellent tool for robot localization as it exists today, there are still some downsides to its implementation. One of the biggest barriers to using the technology is the need to deploy a mesh of anchor antennas accurately located within the environment one would like to track in. This also includes the reality that any new environment will need another UWB anchor mesh installed. This chapter focuses on experiments performed using a new UWB implementation from Decawave using PDoA that does not require an anchor mesh. All that is required are two antennas: one tag to be tracked and one node (anchor) to track the tag. Decawave's PDoA comes out of the box with the ability to track position in 2D and this chapter will show an extension of that position tracking capability into 3D.

### Phase Difference of Arrival Localization [72]

Decawave's PDoA implementation utilizes a single crystal oscillator to seed two phase locked loops (PLLs), as shown in the left of Figure 7.1. This ensures that the two PLLs deliver the same signal for calculation of each node antenna channel's complex response to the incoming tag signal. This hardware setup allows PoA to be calculated for each antenna and thus the PDoA to be calculated between them.

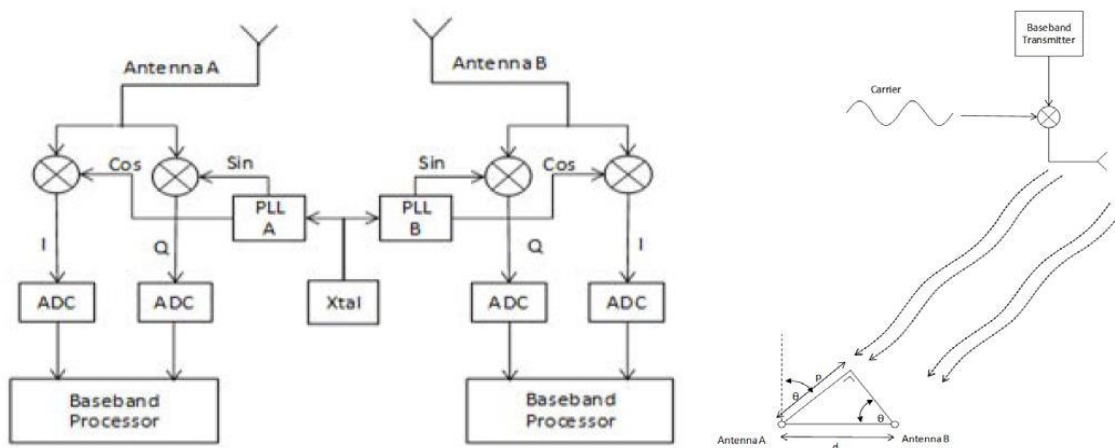


Figure 7.1: Left: Decawave's PDoA hardware implementation. Right: Impinging signal AoA geometry [72].

The right side of Figure 7.1 shows the geometry of how angle of arrival (AoA) is calculated from PDoA. A single signal arriving at an angle,  $\theta$ , to an antenna array will travel a longer path length,  $p$ , to one of the antennas relative to the other. The distance between the antennas,  $d$ , combined with the other two variables leads to the geometric relationship of equation (7.1).

$$p = d \sin(\theta) \quad (7.1)$$

Let the wavelength of the impinging signal be denoted  $\lambda$ . Then the PDoA,  $\phi$ , can be defined as in equation (7.2). This can be verified by substituting values in for the path length equal to zero, half of a wavelength, one whole wavelength, etc.

$$\phi = 2\pi \left( \frac{p}{\lambda} \right) \quad (7.2)$$

Combining equations (7.1) and (7.2) with substitution leads to the relationship of equation (7.3), where the antenna separation distance is still unknown.

$$\theta = \arcsin \left( \frac{\phi \lambda}{2\pi d} \right) \quad (7.3)$$

If the antenna separation distance is constrained to be exactly half a wavelength,  $d = \lambda/2$ , then equation (7.3) simplifies to equation (7.4), which is relationship for AoA that Decawave derives in their paper.

$$\theta = \arcsin \left( \frac{\phi}{\pi} \right) \quad (7.4)$$

Decawave does not use equation (7.4) directly in calculating their results due to a slight deviation in the antenna separation and real-world effects of antenna coupling at such small distances but multiplies eq. (7.4) by a factor of  $\sim 1.05$  to account for these effects. Decawave additionally leverages their existing UWB technology to calculate the ToF between the tag and node. This distance, along with the AoA, allows calculation of 2D position within the plane of the node antenna array. Decawave concludes their paper by alluding to the possibility of 3D position tracking by utilizing more complex antenna arrays. This chapter achieves this possible 3D implementation by simply using two Decawave PDoA nodes arranged side-by-side and perpendicular to each other to measure the two spherical angle values and a distance.

## Theoretical Approach

Although Decawave's whitepaper discusses AoA as a function of PDoA, their beta PDoA demonstration kit outputs (x, y) positions directly from the embedded processor to the serial port. Their demonstration Windows GUI then consumes that information for onscreen display. As our goal is not to simply verify Decawave's analytical approach, raw sensor data for PDoA and distance was pulled from the node instead to facilitate mathematical modeling. Equation (7.5) is the relationship between a general cartesian point,  $x$ , and the variables radius  $r$ , and spherical angles  $\theta_{horz}$  and  $\theta_{vert}$ , that would define its position relative to the origin.

$$x = f(r, \theta_{horz}, \theta_{vert}) \quad (7.5)$$

However, using two nodes as a makeshift 3D antenna array will return four variables, so we will simplify the treatment by assuming that UWB distances are fully 3D already and average the distances from each node to create just one radial distance, using equation (7.6).

$$r = f(r_{horz}, r_{vert}) = \frac{r_{horz} + r_{vert}}{2} \quad (7.6)$$

The nodes return PDoA measurements, not AoA measurements, so we must consider the functional relationships in equations (7.7) and (7.8). Using these equations to define mathematical models will be one approach taken as an intermediate step to solving eq. (7.5). Once the functions are defined, the well-known spherical-to-cartesian trigonometric equations [134] can be used to solve for position.

$$\theta_{horz} = f(\phi_{horz}, \phi_{vert}) \quad (7.7)$$

$$\theta_{vert} = f(\phi_{horz}, \phi_{vert}) \quad (7.8)$$

The other approach investigated will be to use equation (7.9) to find a direct mathematical relationship between individual sensor distance, PDoA, and cartesian position using artificial neural networks (ANNs).

$$x = f(r_{horz}, \phi_{horz}, r_{vert}, \phi_{vert}) \quad (7.9)$$

The approach taken to find these functional relationships is a computational tack, not analytical, as there many unknowns and sources of error not accounted for in these equations.

## PDoA Hardware Introduction

Decawave was kind enough to provide a complementary beta PDoA kit along with an extra node and extra tags to support academic research using their technology. The packaging of said kit is shown below in Figure 7.2.



Figure 7.2: Decawave Beta PDoA kit packaging.

After unboxing, much documentation took place of the novel hardware, the results of which are shown in Figure 7.3 of the fronts and backs of the node and tag modules. Unlike the TREK1000 modules, these development boards can be flashed using a second micro-USB connection to a computer with the correct driver and software installed.

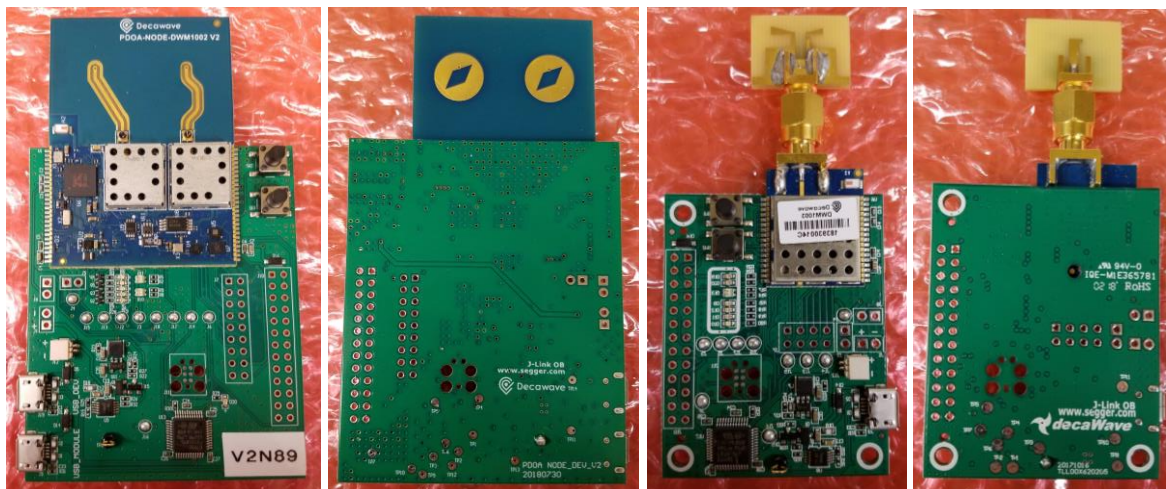
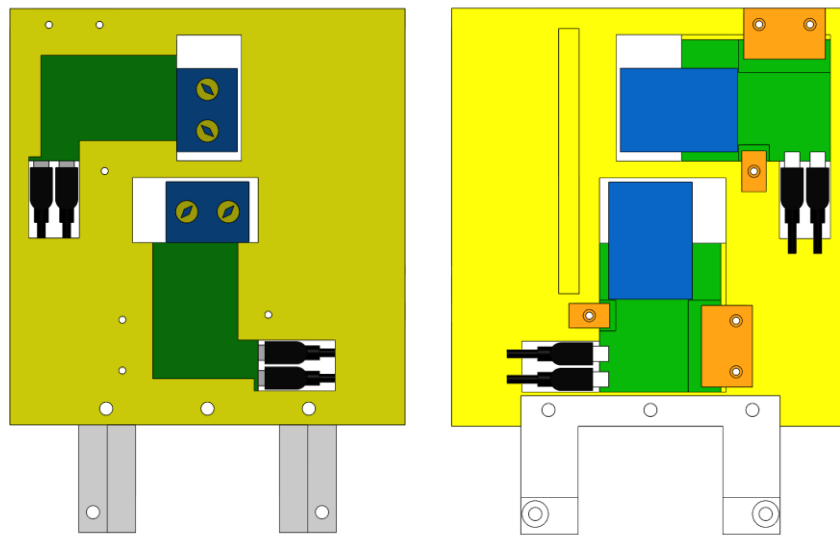


Figure 7.3: From left to right: 1. PDoA node front. 2. PDoA node back. 3. PDoA tag front. 4. PDoA tag back.

## PDoA Shield Design

Two PDoA nodes were obtained with the goal of combining them together into a 3D sensor array. Theoretically, a perpendicular configuration of planar sensors should enable 3D information, given the correct synthesis of each sensor's independent data. The module created by combining the two PDoA nodes into a fixture will be henceforth referred to as the PDoA shield. The CAD for the PDoA shield assembly is shown below in Figure 7.4.



*Figure 7.4: PDoA shield modeled in SolidWorks to align the two PDoA nodes planes as close to intersection as possible. The upper node is termed the vertical node and the lower node is the horizontal node, based on the plane that they operate in.*

The PDoA shield was designed in SolidWorks around the specific dimensions of the Decawave beta PDoA nodes. Care was taken to ensure that there would be space for all USB cables to be connected while the nodes were still mounted in the fixture. Additionally, indexing features were included to ensure mechanical repeatability of all experiments, even after reassembly. The PDoA shield was designed to be mounted in place of the handle on a Trimble S3 total station, but it can also stand freely on its own by utilizing the plastic feet that were included in the demo kit for holding the PDoA tag and node vertical during testing. The fixture was manufactured using 3D printing and was assembled using a few nuts and bolts along with 3D printed circuit board clamps. See pictures of the assembled PDoA shield in Figure 7.5. The CAD files for this assembly can be found in the appendix.



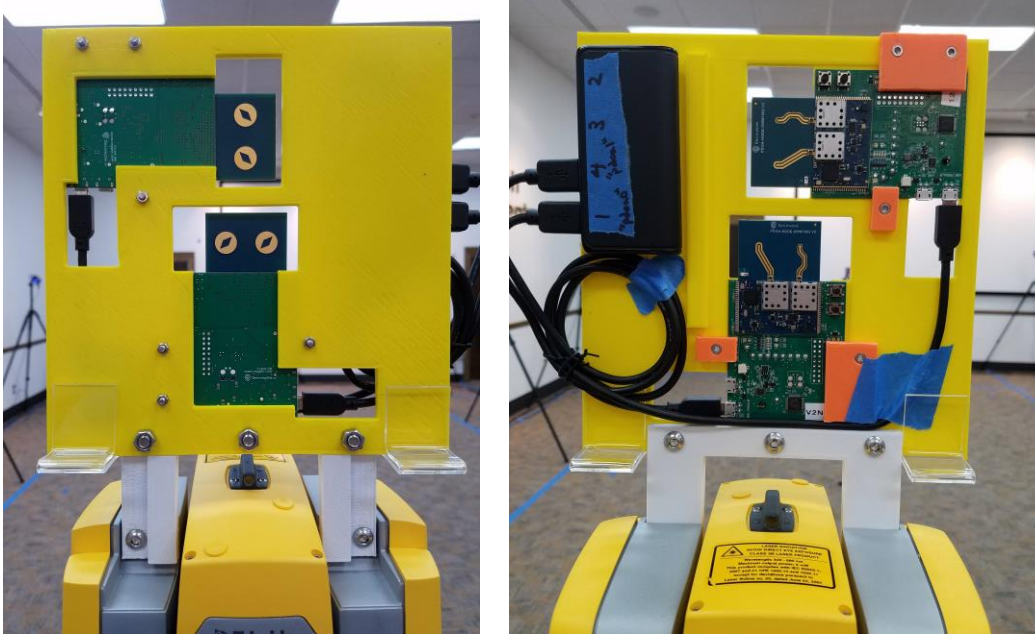


Figure 7.5: PDoA shield assembly with all PDoA hardware bolted to 3D printed plastic and USB cables connected for streaming serial data. The entire shield assembly is connected to the Trimble S3 total station by a mounting bracket.

### PDoA Shield Calibration

Calibration of the PDoA nodes is required before use to initialize the correct linear offset for ranging and PDoA calculation that results from slight differences in manufacturing [135]. The methodology used to achieve consistent calibration deviated from the user manual instructions that directed the user to simply calibrate the node and tag directly inline, facing each other at 3m separation. A hopefully more accurate method is as follows:

- Mount PDoA node in the horizontal opening of the PDoA shield (Figure 7.7, left).
- Mount PDoA shield onto the total station and point it at a wall.
- Use direct reflex (DR) measurement on the total station to point directly at the wall by minimizing measured distance and zero the total station HA here.
- Aim total station directly at the wall at the proper height offset for the PDoA node with laser on (Height offset measurement of 9in shown in Figure 7.6, left).
- Mount PDoA to the wall using a bracket that ensures 10cm of free space between the tag and wall with the center of the antenna in the laser beam (Figure 7.7, right).
- Measure the actual antenna position and adjust until the tag antenna is directly in front of the PDoA node at the same height as the horizontal node (Figure 7.8).

- Run the calibration routine in the included PC GUI executable with the measured distance entered. Include offset from PDoA mounting on total station (Figure 7.6, right).
- Repeat for the vertical node, also placed in the horizontal window for consistency.
- Both nodes are now calibrated for their measurement plane and ready to be assembled for experimental data collection.

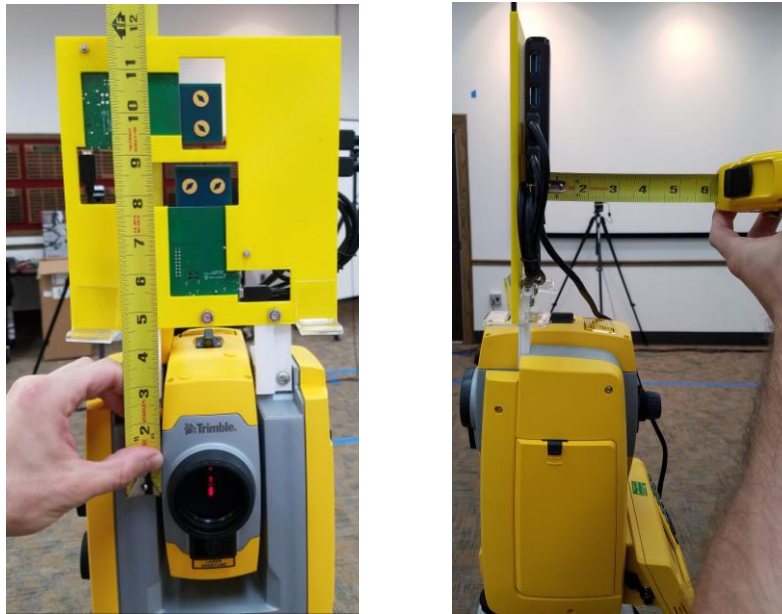


Figure 7.6: Measuring PDoA shield offsets from the total station origin. Left: height offset. Right: distance offset.

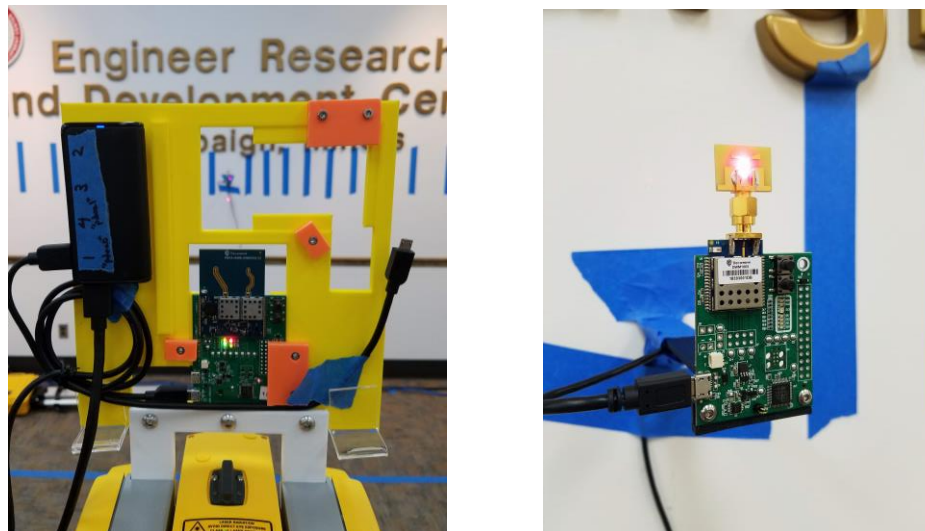


Figure 7.7: Calibrating the PDoA nodes for distance correction and PDoA offset constant.



Figure 7.8: Left: Lining up the PDoA tag with the total station zero elevation and zero easting. Right: Offsetting the PDoA tag with the mounted PDoA node height above the total station for PDoA calibration.

### PDoA Driver Code Modification

The existing PDoA node driver code was kept as exactly shipped to the greatest extent possible. However, the stock driver reports calculated Cartesian coordinates (x, y) in the plane rather than the raw distance and PDoA measurements desired. The driver code was therefore modified using the source code provided with the kit and then re-flashed onto the nodes using the integrated JTAG circuitry. The modifications made were adding a function to calculate the offset corrected PDoA onboard and wrap this to  $\pm 180$  degrees, increase precision in the output distance, and publish offset corrected PDoA in the level 2 and 3 serial message reports. Level 1 reports were not modified, as these are used by the PC GUI executable for calibration and graphic functionality checks. The units of the variables were changed to millimeters and millidegrees, because there was space in the variable types to do so and still maintain full sensor range. The only source code file modified was json2pc.c and a summary of the changes is as follows:

- Calculate offset corrected PDoA on board and wrapTo180 (domain of  $[-180, 180]$  degrees).
- Changed output data from calculated (x, y) to distance in mm and corrected PDoA in mDeg.
- Print format changed from unsigned hex integer to signed decimal integer of length 8:
  - Max distance = 999999999mm = 100 km (units now an order of magnitude more precise, in millimeters, than UWB's centimeter level accuracy).
  - -180deg = -0180000mdeg (large enough to handle complete possible PDoA range).
- Report levels 2 and 3 modified as listed, level 1 is used by the PC GUI and was not changed.

### First Experiment on the Wall

A first pass at experimentation was attempted using the setup shown in Figure 7.9. The PDoA tag was affixed to a bracket on the wall sequentially at each of the tape markers. Then, the total station was turned to point at the tag antenna to measure its location on the wall before being turned back to point at ( $HA = 0$ ,  $VA = 0$ ). At this point, each PDoA node was manually actuated one at a time using the TeraTerm terminal emulator [136]. About 30s was timed for each actuation to collect data before moving the tag to the next position. Terminal output was automatically saved in a text file for each node. Both nodes use the same radio frequency and airspace algorithm, so they will collide if run at the same time. This necessitates a system that switches back and forth between which node is on and prevents real-time 3D positioning.



*Figure 7.9: Experimental setup for analyzing PDoA measurements constrained to a single plane.*

After a great deal of manual data cleaning and organizing, a small set of data was available to plot in MATLAB, which can be seen in Figure 7.10. These plots show horizontal and vertical PDoA measurements, plotted on the Z-axes versus position on the X-axis and Y-axis, respectively, as defined in Figure 7.9. There is not too much information to be gleaned from this experiment other than that the PDoA system seems to work, and the vertical node has noisier measurements than the horizontal node. This greater noise would make sense at the extremes of the Y-axis, due to out-of-range use, but noise seems to correspond with the Z-axis, in which direction this node should operate best.



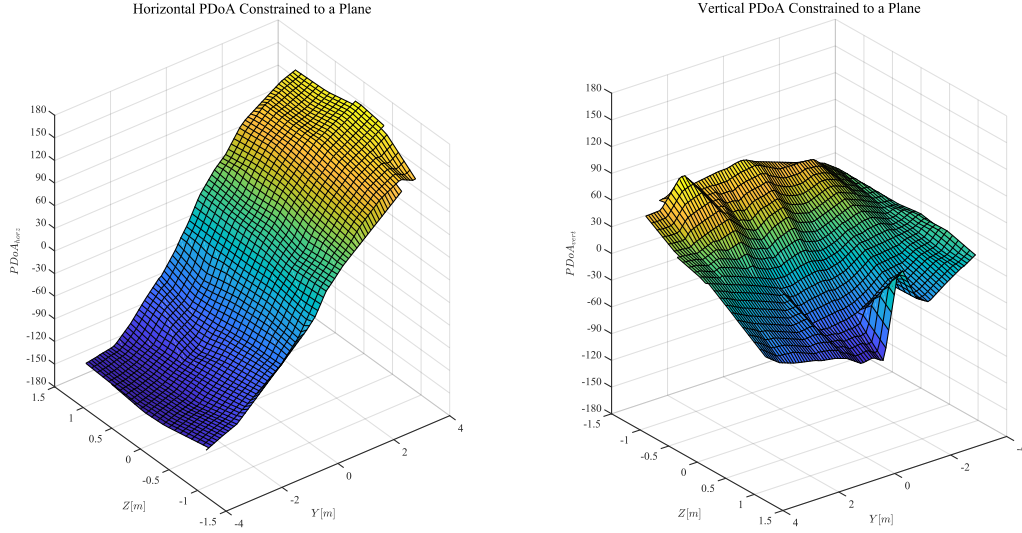


Figure 7.10: PDmA data collected when constrained to a single plane. Left: Horizontal PDmA measurements Right: Vertical PDmA measurements. Vertical data is much noisier than horizontal data. Note that the views are different in each plot to maximize feature visibility.

## ROS PDmA Driver

One of the biggest takeaways from the first experiment was that any further investigations would have to be automated in terms of data collection and interfaced with the Vicon MoCap system to provide ground truth pose data. To this end, ROS was again used as a foundation to build a serial driver and approximate time filter upon. Due to the requirement switching operation so as not to interfere between the nodes, the data structure of Figure 7.11 was used to collect equal measurements from each PDmA node and publish a leading line of zeros for distinguishing location movements of the PDmA tag in MATLAB.

PDmA Time	PDmA Horz	PDmA Vert	Publish	Vicon	Output
t0	0	0	1		(0,0,VICON)
t1	PDmA_Horz1	-	0		-
t2	PDmA_Horz2	-	0		-
t3	PDmA_Horz3	-	0		-
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
t198	PDmA_Horz198	-	0		-
t199	PDmA_Horz199	-	0		-
t200	PDmA_Horz200	-	0		-
t201	PDmA_Horz1	PDmA_Vert1	1		(Horz1,Vert1,VICON)
t202	PDmA_Horz2	PDmA_Vert2	1		(Horz2,Vert2,VICON)
t203	PDmA_Horz3	PDmA_Vert3	1		(Horz3,Vert3,VICON)
⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮
t398	PDmA_Horz198	PDmA_Vert198	1		(Horz198,Vert198,VICON)
t399	PDmA_Horz199	PDmA_Vert199	1		(Horz199,Vert199,VICON)
t400	PDmA_Horz200	PDmA_Vert200	1		(Horz200,Vert200,VICON)

Figure 7.11: PDmA driver data custom ROS message structure. Vicon is shown as grey because it is comparably dense in time.

### *Serial Node*

The main functionality of the ROS PDoA driver serial node is to pull serial messages from the output buffers of the PDoA nodes and parse them in real-time for data collection. In the future, it will also provide the framework for creating a real-time implementation of a localization engine. The driver works by first initializing each of PDoA node serial ports and setting the serial reporting to level 3 (the shortest messages). Both nodes are put into the "stop" mode to be ready to start when called. The driver then enters a continuous loop that is 1000 times faster than 10Hz frequency that the PDoA nodes publish at.

The two PDoA nodes interfere if running simultaneously, therefore a switching method is used to first turn on and collect a set of data from one node, which is then turned off and the corresponding set of measurements is also collected with the second node. The nodes do not switch modes quickly, which rules out the possibility of rapidly switching them on and off in alternation. Each time the horizontal node outputs data in the serial buffer, the driver immediately timestamps it, parses the data, and either rejects it as error or accepts the message and stores it in a temporary array. Once the temporary array has been filled to the desired capacity, the exact same process repeats for the vertical node except that instead of storing each message, they are published from the temporary array in a tuple along with the corresponding horizontal data. Once the desired number of datapoints have been published, the driver waits for the user to press the enter key to repeat the loop.

Once a set of datapoints has been collected, the operator may move the PDoA shield or tag before collecting more data. The temporary data array length used in these experiments was 200 points, which leads to about 40s of data collection per iteration of the algorithm. Currently the driver only supports tracking one tag, but this could be easily extended based on the unique addresses of each tag. Because of the required node switching, it is necessary for a tag to stay static throughout the duration of each loop if each PDoA node's data is to be meaningfully correlated. The algorithm structure is summarized below:

1. Initialize publisher ROS node.
2. Initialize horizontal PDoA node serial connection, change reports to level 3, and place into "stop" mode.
3. Initialize vertical PDoA node serial connection, change reports to level 3, and place into "stop" mode.

4. Place horizontal node into "node" mode, read, timestamp, and parse serial messages, place them into a temporary array of the desired length, and put the node into "stop" mode once the array is full.
5. Place vertical node into "node" mode, read, timestamp, and parse serial messages, publish them along with the horizontal array contents in corresponding order as tuples, and put the node into "stop" mode once the array length has been fully read.
6. Wait for user to adjust physical setup and press "Enter" to repeat.

### *Approximate Time Message Filter*

ROS provides a set of filters for sorting messages based on their timestamps. When using Vicon MoCap for ground truth, it is extremely useful to be able to line up the studied topic in time with Vicon for direct point-by-point comparison. Vicon publishes at a rate much faster (up to 200Hz) than the PDoA nodes publish (10Hz). This theoretically guarantees that every PDoA data point will have a unique ground truth datapoint closest to it in time. The approximate time filter operates this matching procedure by choosing the messages closest in time to each other. Once the PDoA and Vicon messages match with each other, a callback function is executed, which in this case simply concatenates Vicon pose data with PDoA data tuples in the custom ROS message shown in Figure 7.11. The full (PDoA data, Vicon ground truth) tuples are then published on their own ROS topic, which is recorded using rosbag for later offline analysis in MATLAB. Due to the length of these multiple-hour-long experiments, the raw Vicon ground truth topic was not recorded, to keep bag sizes manageable.

### *Data Analysis*

The ROS package in MATLAB was used to parse ROS bag files into manageable matrices and arrays. To accomplish this, the custom PDoA ROS message had to be imported into MATLAB, but there is a documented process to do so [137]. Once imported into MATLAB, the sensor data was manually adjusted for analysis in standard units and coordinate conventions. The raw PDoA measurements were converted from millidegrees to degrees and radians. The raw distance data were converted from millimeters to meters. The MATLAB spherical coordinate convention is shown in Figure 7.12 and adhering to this required reversing the sign on the vertical node PDoA, as will be discussed in the next section. Next, the raw ROS data was segmented for point-by-point analysis in space by dividing the raw data list based on where all leading zeros were published to indicate a new point. Finally, plotting and modeling was undertaken with the goal



of finding candidate functions to convert raw sensor data from the PDoA shield into 3D cartesian point positions for the PDoA tag.

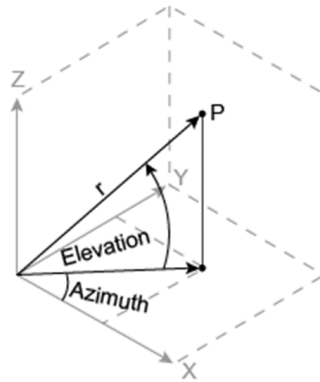


Figure 7.12: MATLAB spherical coordinate convention [134].

### Experimental Setup

The remainder of the PDoA experiments were conducted in the Vicon MoCap arena, with the PDoA shield near the arena's center, as shown in Figure 7.13. There are several different coordinate frames at play, so it would be best to first define them all and illustrate their relationships. Figure 7.13 shows the Vicon MoCap coordinate frame, as consistent with the UWB experiments in previous chapters. Although this was convenient to reuse at the time, it would have been better to rotate this frame 90 degrees about the Z-axis to make it consistent with the PDoA shield frame, which is shown in relation to the Vicon frame in Figure 7.14.

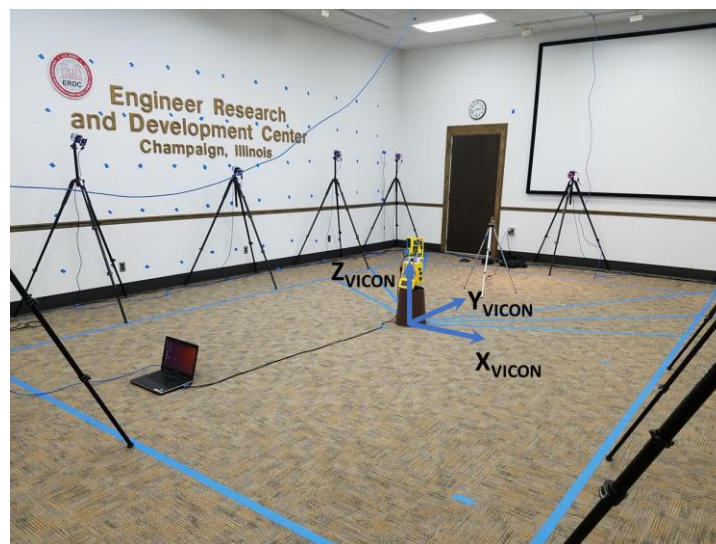


Figure 7.13: Vicon is hardwired into the computer that is collecting the PDoA data. The frame origin is located on the floor.

To remain consistent with previous PDoA results from Decawave and to stay symmetric about the zero AoA definition of a signal impinging perpendicularly to the antenna array, the PDoA shield frame was chosen as shown in Figure 7.14.

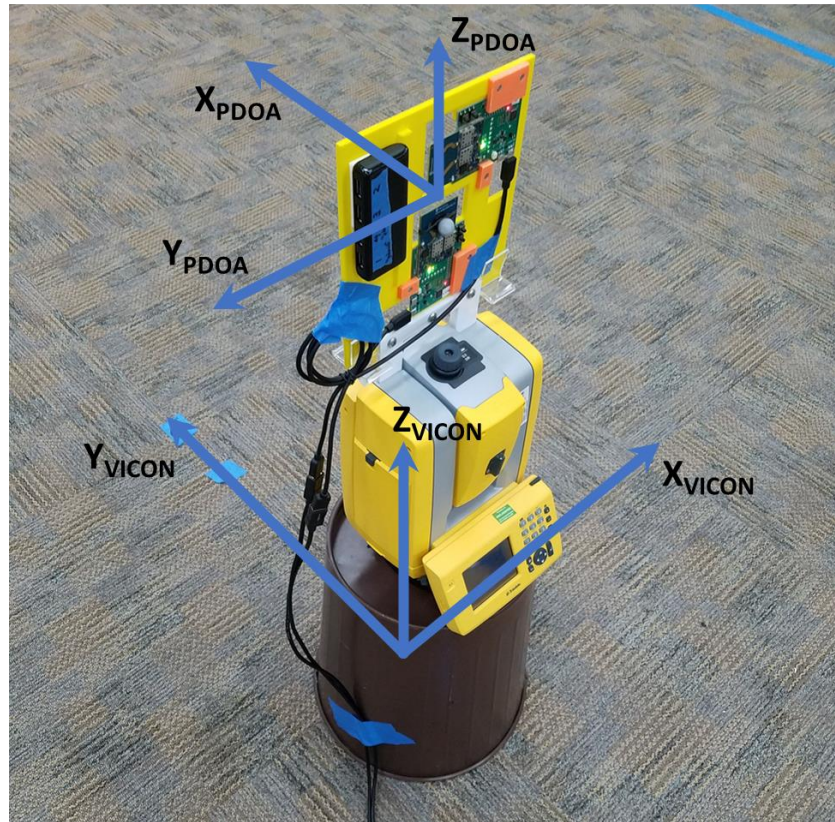


Figure 7.14: Vicon and PDoA frames are rotated 90 deg from each other in the plane. This must be compensated for in the data analysis.  $X_{PDoA}$  is normal to the planar antenna array.

To remain consistent with this PDoA frame definition and the MATLAB spherical coordinate convention, the individual PDoA node frames had to be accounted for. They are shown in Figure 7.15 and while the horizontal node is consistent with the PDoA shield frame, the vertical node is not. The shield was designed to optimize cable management, otherwise the vertical node USB cable would stick into the air if the node were rotated 180 degrees. Instead, the vertical node PDoA data was simply reversed in sign in post-processing to flip the positive angle to be upwards. This frame transformation could be hidden elsewhere, such as in the PDoA node code, but the math to be done in data analysis is not difficult and it was desired to maintain source code symmetry between the two PDoA nodes.

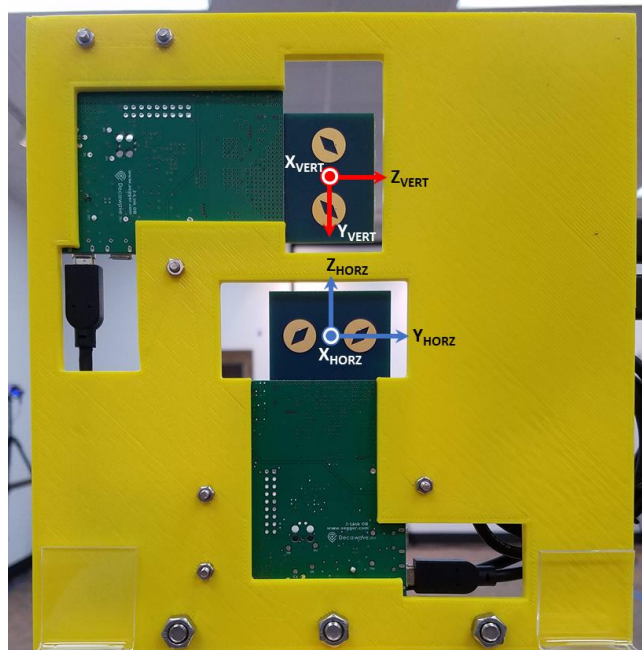


Figure 7.15: This setup is called the PDoA shield. Each individual board is called a PDoA node. Vertical PDoA node is oriented such that positive angles are down, and negative angles are up. This needs to be inverted in data analysis to be consistent with the spherical coordinate convention. Both nodes cannot operate simultaneously as they will jam each other's channel.

The PDoA shield and tag both needed Vicon Tracker objects to ensure their ground truth origins were coincident with their sensor reporting positions. Additionally, the PDoA shield Vicon object frame needed to be consistent with its physical definition. The MoCap marker arrangement to achieve this is shown in Figure 7.16. First, the total station was leveled and left on, to make sure that the connecting line between the two node origins was parallel with gravity and the Vicon Z-axis, which is also set to gravity via bubble level on the Active Wand. Next, the object origin and orientation were manipulated in Vicon Tracker to be centered between the two nodes and consistent with the definition of the PDoA shield frame in Figure 7.14. Finally, the PDoA shield origin and X-axis were physically aligned with the Vicon frame as shown in Figure 7.17. The rotation and position of the PDoA shield were manually manipulated until they were aligned with the Vicon frame as seen at full zoom of the Tracker program. The Trimble S3 total station was kept on throughout the experiments to utilize its motors to hold the PDoA shield rigidly locked in this orientation. A more accurate way to compensate for misalignment would have been to simply transform the ground truth data in analysis, but it was not desired at the time to add another element to further complicate the analysis.



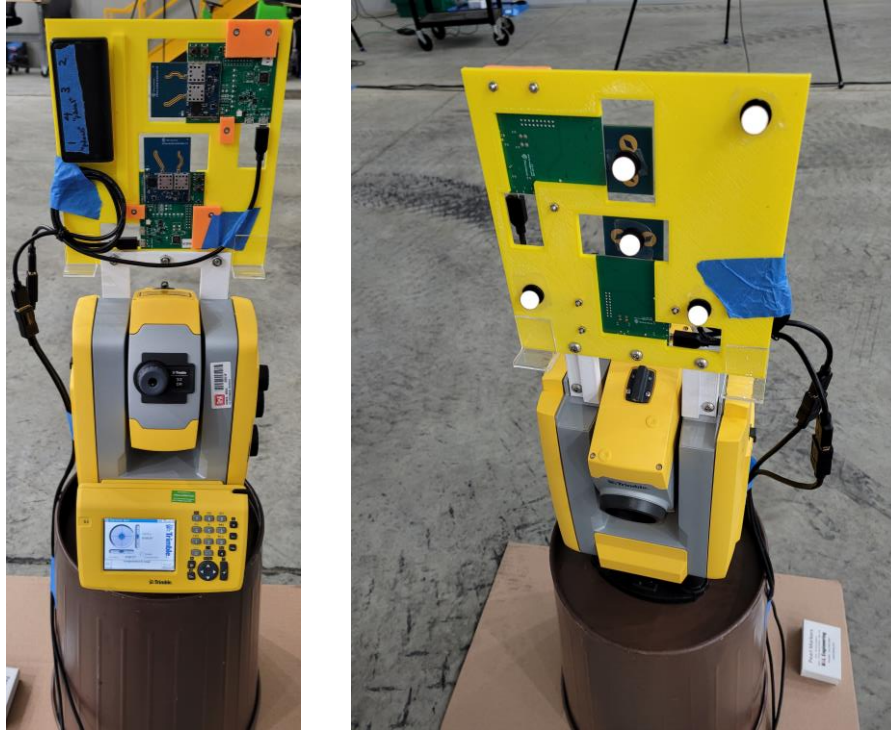


Figure 7.16: Left: Leveling the PDoA shield using the total station. Right: Centering the Vicon object PDoA origin to the origin of the PDoA shield. Not shown are two additional MoCap markers on the back of each PDoA node.

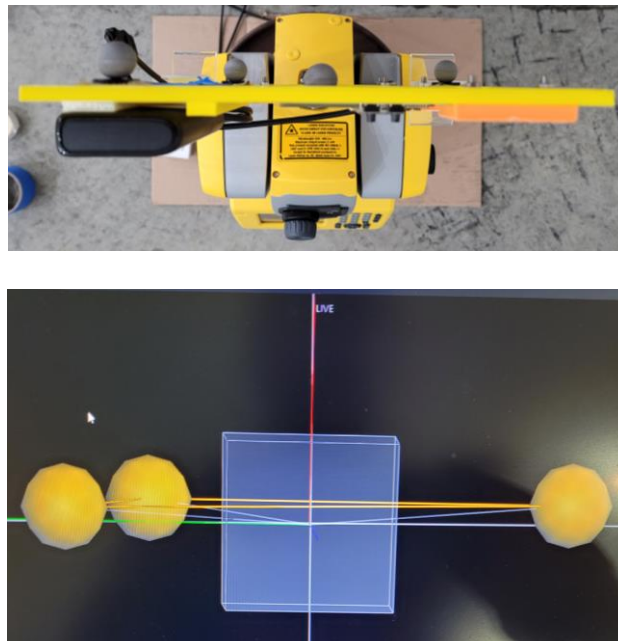


Figure 7.17: Aligning the PDoA x-axis with the MoCap system. The total station remains powered throughout the experiment to maintain PDoA X-axis alignment.

The PDoA tag Vicon Tracker object was initialized in a similar fashion to the PDoA shield, but with only the requirement that the origin be aligned. This process is shown in Figure 7.18,

where first the origin is snapped to reside between the two markers attached to the antenna, before removing these markers from the Vicon Tracker object and leaving the origin to be coincident with the antenna. All markers must be removed from all antennas before use so as not to affect their radio performance.

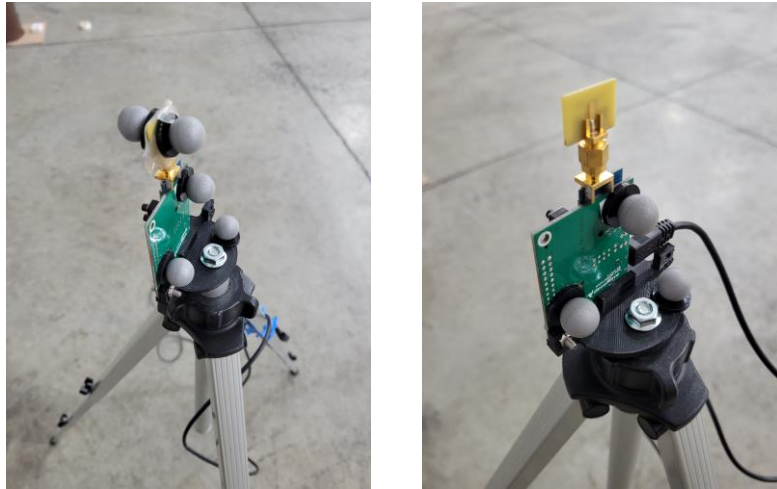


Figure 7.18: Left: Creating the PDoA tag origin in MoCap. Right: The remaining three retroreflectors used to locate the tag.

Figure 7.19 below shows the masking tape used on the floor of the Vicon arena to keep track of AoA values for the PDoA shield while moving the PDoA tag about the arena on a tripod. The tripod height was adjustable to allow enough DoF for full 3D variation of PDoA tag location during data collection.

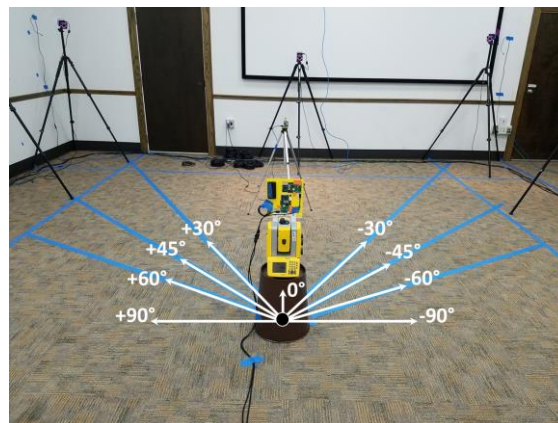
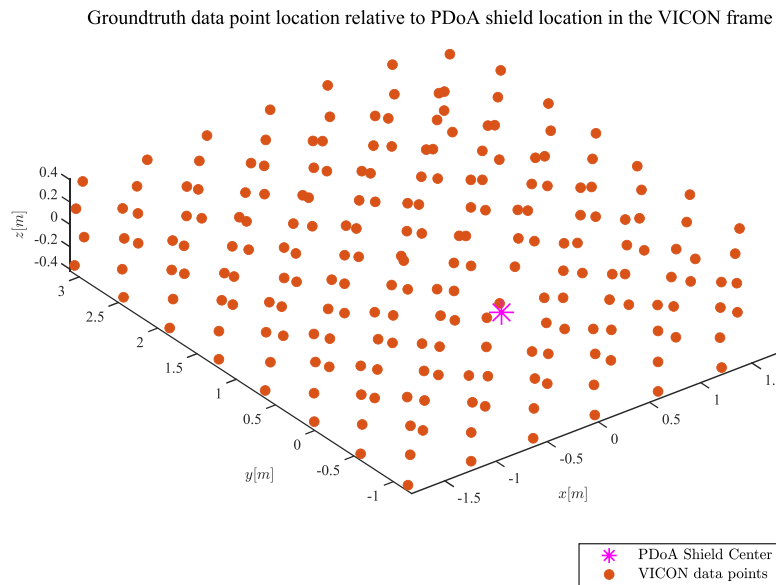


Figure 7.19: Tape on the floor of the MoCap arena indicating horizontal angle. Horizontal angles are measured relative to the PDoA frame.

## Results

### Data Introduction

The PDoA tag locations that data were collected at are plotted in Figure 7.20. These data are location averaged for visualization and there are a total of 220 points in the dataset in a grid of count 7 x 8 x 4, minus 4 points where the PDoA shield was positioned.



The spherical coordinate locations of these points relative to the PDoA node are shown in Figure 7.21 as a visualization of what angular measurement ranges of data were collected. The figure shows that horizontal PDoA data was collected in a full circle around the origin. Although it was known before the data collection that the PDoA nodes only had a range of  $\pm 90$  degrees due to the one-sided directionality of the PDoA node antenna, this was verified by collecting 25% of the measurements behind the PDoA shield. Figure 7.22 shows how the range of vertical PDoA data that could be captured was much more limited by the height of the Vicon cameras. Most of the vertical data was captured within  $\pm 15$  degrees, with some of the data extending into  $\pm 30$  degrees vertical. Further research in this topic should strive to achieve a larger range of vertical angle data and all data at greater distances.





Figure 7.21: Horizontal angle and radius of data collection points compared to the PDoA Shield origin.

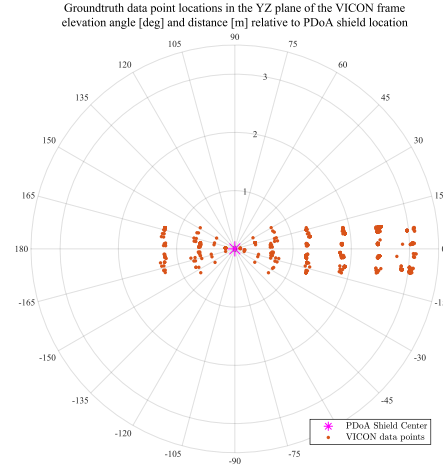


Figure 7.22: Vertical angle and radius of data collection points compared to the PDoA Shield origin.

## Raw Data

Before digging into the results of the mathematical models, it is instructive to first visualize the raw data collected in several ways to define operational range restrictions and gain insight into which mathematical models might best apply to the data. The plots in Figure 7.23 both show horizontal node PDoA data vs horizontal AoA as collected by the ground truth. We can see that the functional relationship between the two variables breaks down outside of roughly  $[-90, 90]$  degrees of AoA range, as is expected due to the antenna.

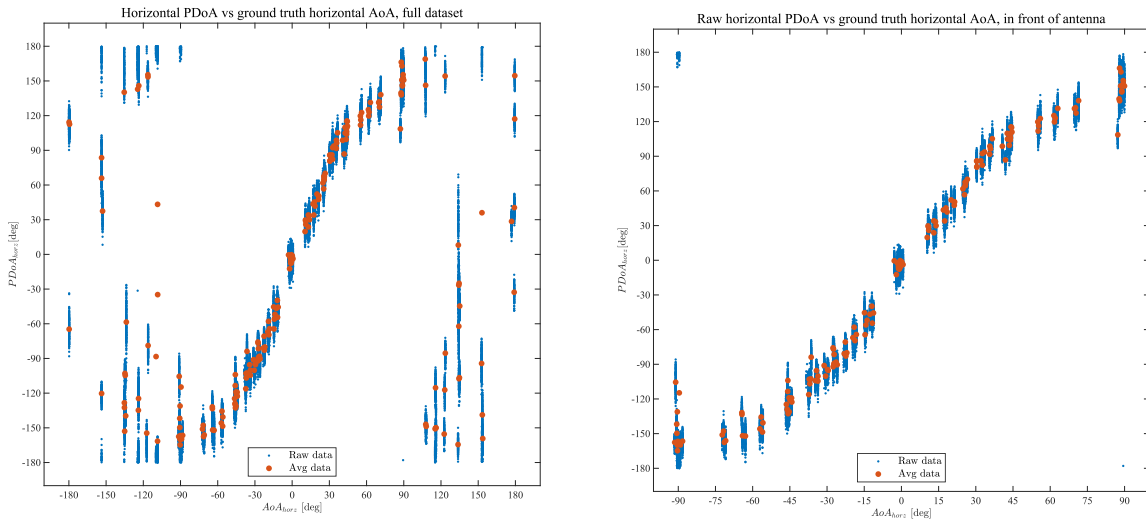


Figure 7.23: Raw and averaged data point visualization of horizontal PDoA vs horizontal AoA. Left: Full dataset from  $-180$  to  $180$  degrees AoA. Right: Same data with horizontal AoA restricted to  $[-90, 90]$  degrees. The functional relationship between PDoA and AoA breaks down around  $\pm 90$  degrees, as is expected from the PDoA node antenna.

These results for horizontal PDoA vs AoA closely reflect the results that Decawave obtained in their whitepaper explaining the technology [72]. The working assumption is that the vertical AoA range does not exceed the operational limits of the horizontal PDoA node. This assumption will not work for the vertical PDoA node. Figure 7.24 shows raw and averaged vertical PDoA data vs vertical AoA, in the same way as the previous horizontal discussion. This time, there is not enough data to know when the functional relationship breaks down, but it should not be until outside of the vertical AoA range  $[-90, 90]$ . Observing the data, it appears much noisier than that of the horizontal node.

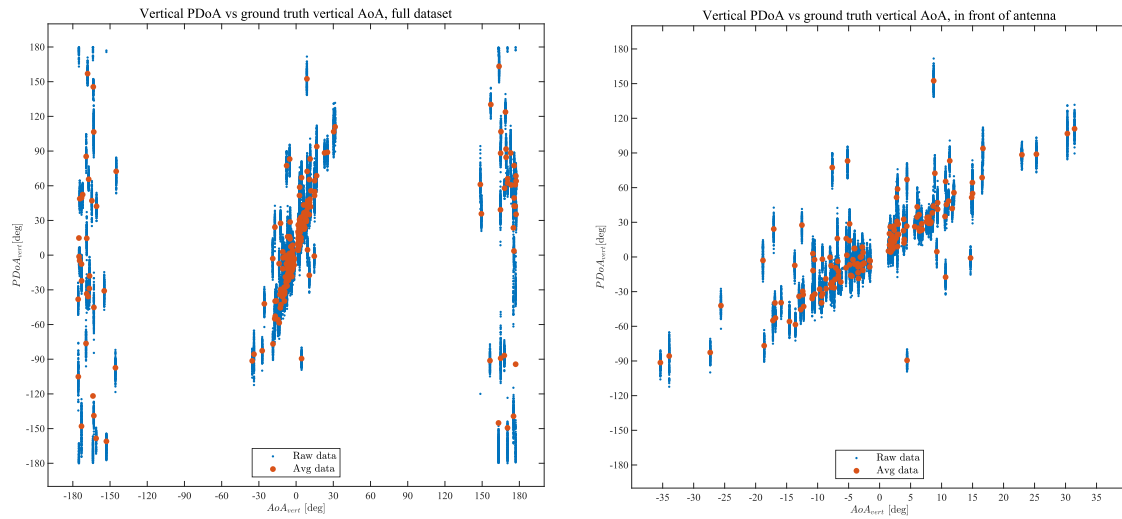


Figure 7.24: Raw and averaged data point visualization of vertical PDoA vs vertical AoA. Left: Full dataset from  $-180$  to  $180$  degrees AoA. Right: Same data with vertical AoA restricted to  $[-40, 40]$  degrees. The functional relationship between PDoA and AoA breaks down behind the PDoA shield, as is expected from the PDoA node antenna.

In Figure 7.25, the vertical PDoA vs vertical AoA data has been colored based on the range of horizontal AoA that it was collected within. From this visualization, it can be readily seen that outside of a horizontal AoA range of at least  $[-75, 75]$  degrees that the functional relationship between vertical PDoA and vertical AoA suffers. Figure 7.26 further compares horizontal AoA ranges using a linear fit that shows how outside of  $[-60, 60]$  degrees horizontal AoA, the vertical PDoA lies further from the line of best fit. Ultimately, it was decided that the working range of the vertical PDoA node was within the horizontal AoA range  $[-60, 60]$  degrees. Due to sensor symmetry, the horizontal PDoA node range should also be constrained to within a vertical AoA range of  $[-60, 60]$ , however this assumption could not be tested here due to the limited height of the available Vicon system. The assumption should be tested fully in future work.

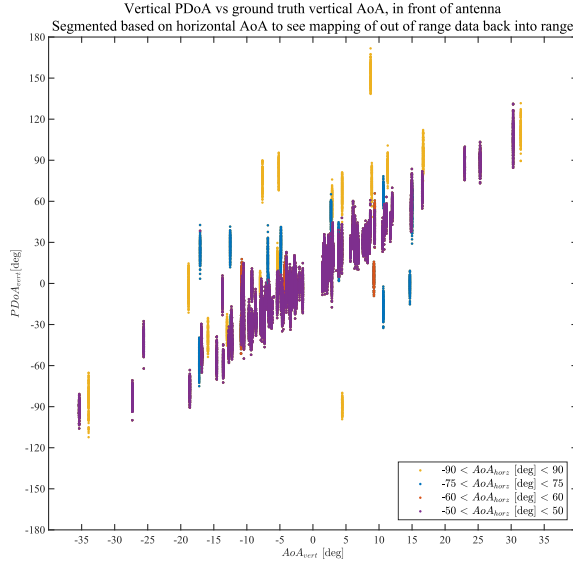


Figure 7.25: Segmented raw data point visualization of vertical PDoA vs vertical AoA for vertical AoA range of  $[-40, 40]$ . Data is color segmented based on the horizontal AoA range that it was captured within.

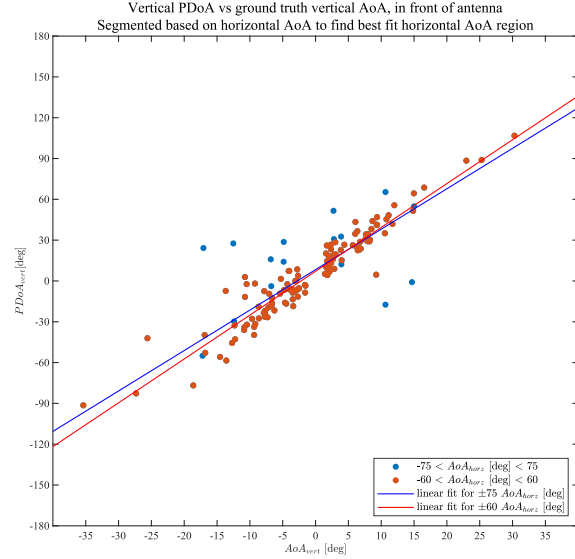


Figure 7.26: Segmented averaged data point visualization of vertical PDoA vs vertical AoA for vertical AoA range of  $[-40, 40]$ . Data compares the horizontal range of  $[-75, 75]$  to  $[-60, 60]$ , along with linear best fit lines for each dataset.

Figure 7.27 below shows the final pared down data sets in the range of  $[-60, 60]$  degrees AoA for each of the PDoA nodes, on axes of the same scale. Before looking at models, let us look at still a few more visualizations of data relationships.

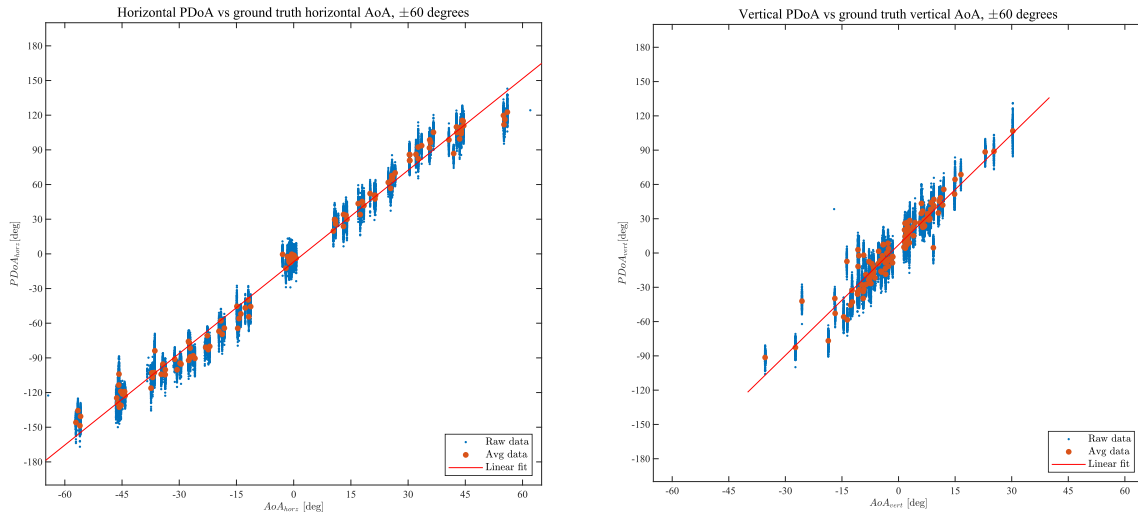


Figure 7.27: Left: Raw and averaged data point visualization of horizontal PDoA vs horizontal AoA for AoA range of  $[-60, 60]$ . Right: Raw and averaged data point visualization of vertical PDoA vs vertical AoA for horizontal AoA range of  $[-60, 60]$ . The PDoA nodes should theoretically give the same data trends in each plot, but the range was not available to fully test this.

The next two plot pairs, Figure 7.28 and Figure 7.29, show the cross relationship between one plane's PDoA node data and the other plane's AoA, segmented by the horizontal AoA range. It

is difficult to glean much information from these plots other than that the nodes do not work well outside of  $\pm 90$  degrees AoA, but we already knew that. At the very least, they are visual 2D projections of the surfaces that we will fit models to in the next section.

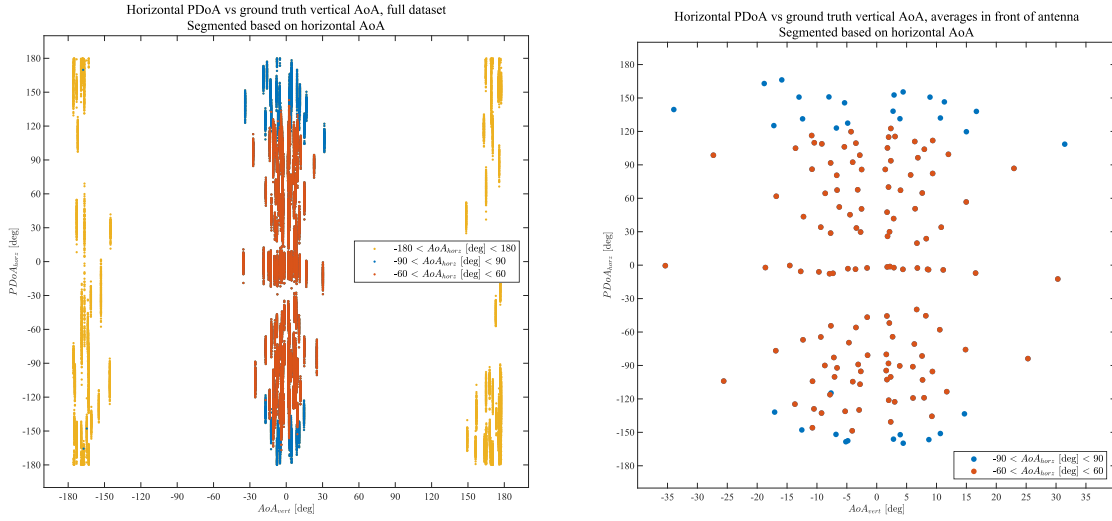


Figure 7.28: Left: Raw data comparison of horizontal PDoA vs vertical AoA, segmented by color based on horizontal AoA range. Right: Same data averaged by collection location and zoomed in on the range taken in front of the PDoA shield.

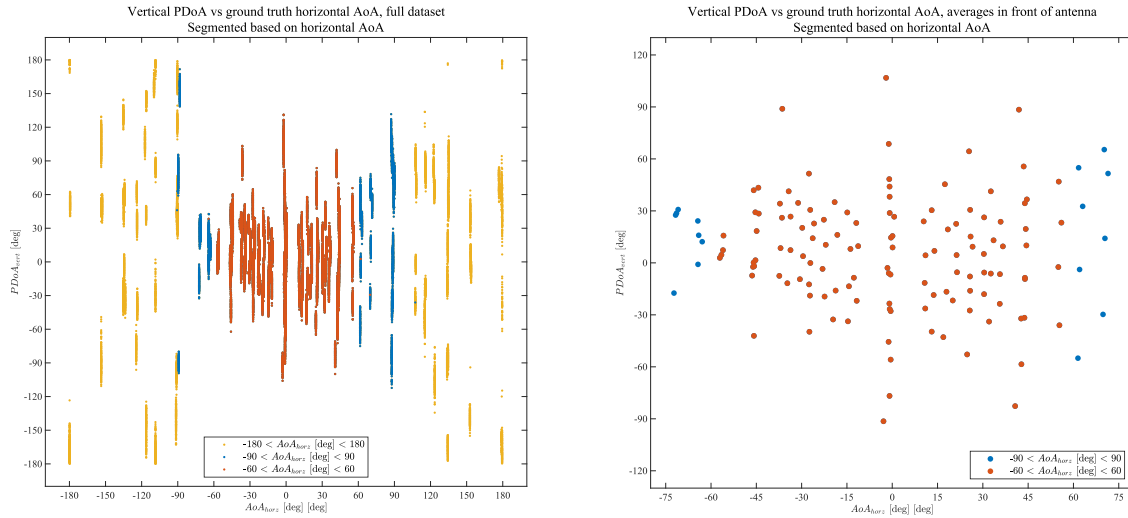


Figure 7.29: Left: Raw data comparison of vertical PDoA vs horizontal AoA, segmented by color based on horizontal AoA range. Right: Same data averaged by collection location and zoomed in on the range taken in front of the PDoA shield.

The last raw data visualization to be considered is shown below in Figure 7.30. This is a novel way of graphically representing the full raw dataset that we are considering. PDoA data are plotted on the X and Y axes, while averaged UWB distance between the two PDoA nodes is plotted on the Z-axis. The result is the shape of the raw data that our goal is to map to the

cartesian points as seen in Figure 7.20. The plot on the left shows the full raw dataset segmented by color into two sets of in range and out of range data. The plot on the right shows only the data points that will be attempted to fit a model to, averaged based on the PDoA tag location that they were collected at. The goal is to find a unique one-to-one mapping from this raw dataset onto the set of cartesian points.

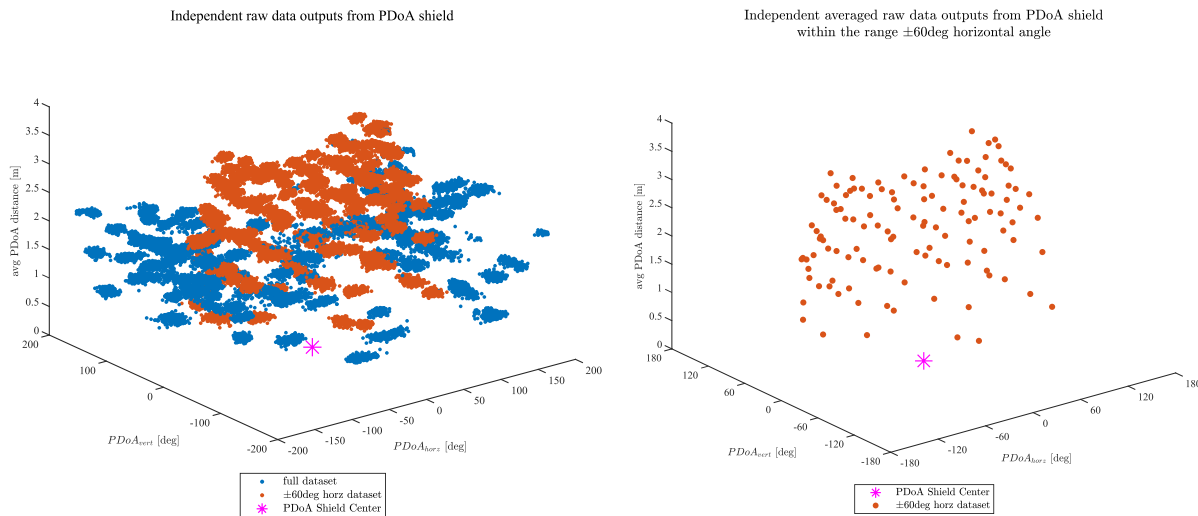


Figure 7.30: Left: raw data visualization of all three independent variables from the PDoA shield into 3D space, segmented by color on whether the data will be included in a mathematical model or not. Orange color points will be included. Right: same data averaged by PDoA tag collection location and showing only the dataset included for modeling purposes.

## Model Fitting

The three models to be discussed here are two surface fits created using MATLAB's curve fitting tool (cftool) and an ANN created using MATLAB's neural network fitting tool (nftool). The intuition behind these methods was that there is clearly a relationship between the PDoA shield's raw data and the PDoA tag ground truth locations. It is certainly non-linear, as we already know from the trigonometric functions involved. There is also a good deal of noise to be seen in the data from many non-idealized sources that are not accounted for. Thus, it would be desirable to have a model that simply transforms raw PDoA data into cartesian locations as somewhat of a black box and remain robust to variations of the input data.

The first model considered was a shallow ANN with 50 hidden layers. A shallow neural network was used to avoid overfitting data to create a robust model, compared to a deep network that would be too complex for this simple function fitting exercise. The first naïve attempt taken was

to train an ANN on the full dataset, not restricted to  $[-60, 60]$  degrees AoA to see if it would just work. The training results are shown in Figure 7.31. In the upper right plot, the trained fit can be seen for each of the four input variables to vary widely from the regression. Already, it appears that this model will not be a good fit.

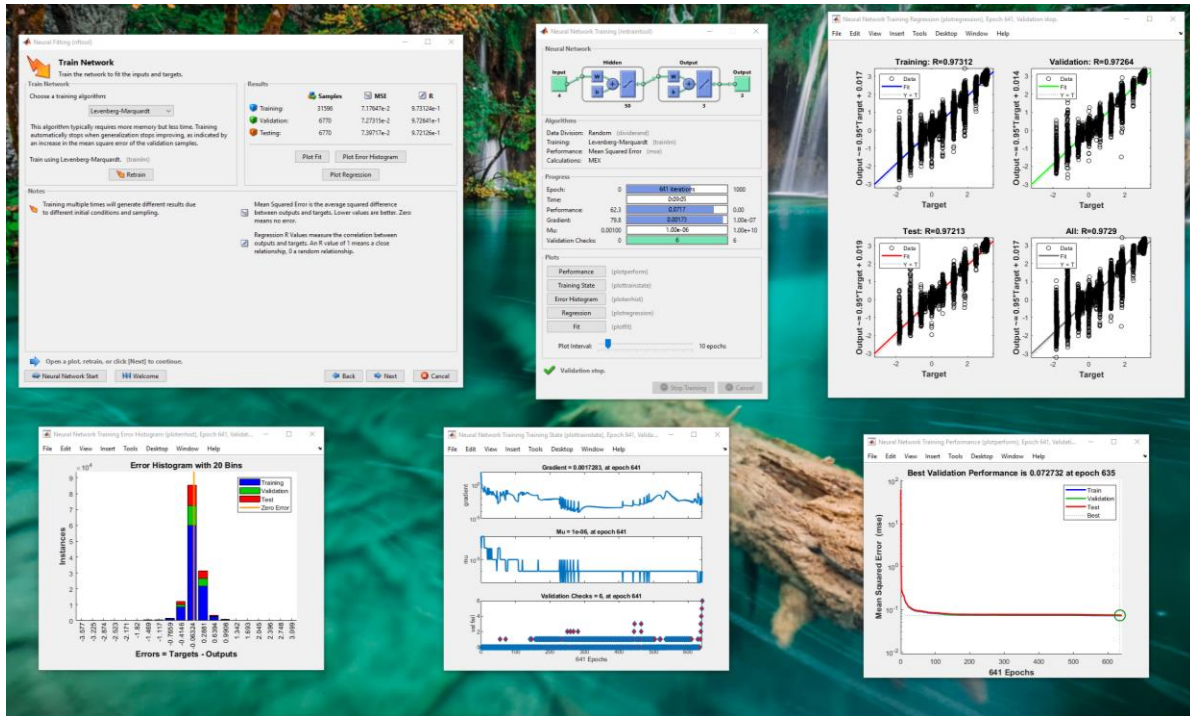
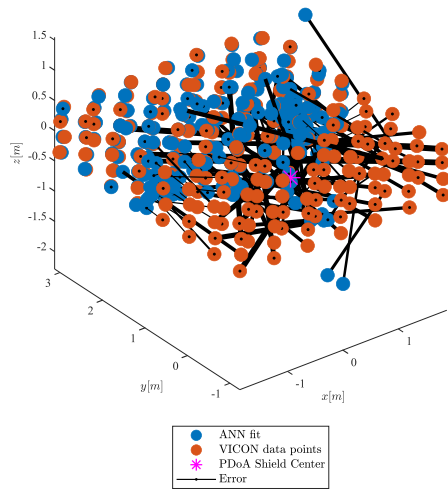


Figure 7.31: Training statistics for a 50 hidden layer ANN using four variable input for the entire ECR dataset.

This can be readily verified in Figure 7.32, which shows a 3D plot of regressed locations versus actual locations, where the blue points are the average of the pointwise ANN fit and the orange points are the location averaged PDoA tag points. Although it appears that the data inside of our bounds does find a good fit, it is apparent after taking these points away, in the plot on the right, that all the data taken from behind the PDoA shield has been mapped by the ANN either in front of the PDoA shield or to a wrong location. This confirms our earlier data analysis in identifying horizontal AoA ranges where sensor data did not have a functional relationship. A shallow ANN can approximate any function with arbitrary precision, but that function must still have a one-to-one mapping from input to output to define a unique relationship.



ANN positional fit of full raw dataset vs ground truth locations



ANN positional fit of full dataset without  $\pm 60$ deg dataset vs ground truth locations

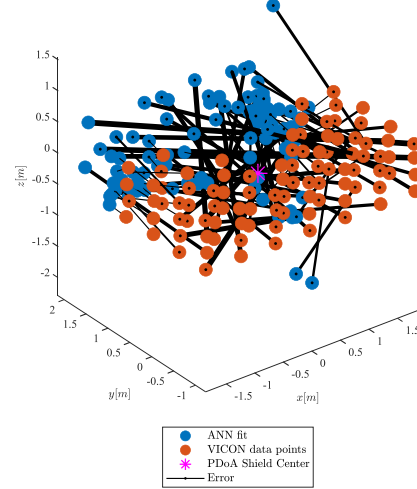


Figure 7.32: Performance of an ANN trained on the full dataset. Left: ANN on full dataset. Right: ANN on out of bounds data.

Continuing with more knowledge, we now astutely remove all the sensor data that falls outside of the AoA range of  $[-60, 60]$  degrees and try training another ANN. Figure 7.33 shows the training data with the same 50-layer shallow ANN configuration. The upper right plot shows that this time the variable regressions are much tighter. Additionally, it took roughly half the number of epochs to converge to an acceptable performance level.

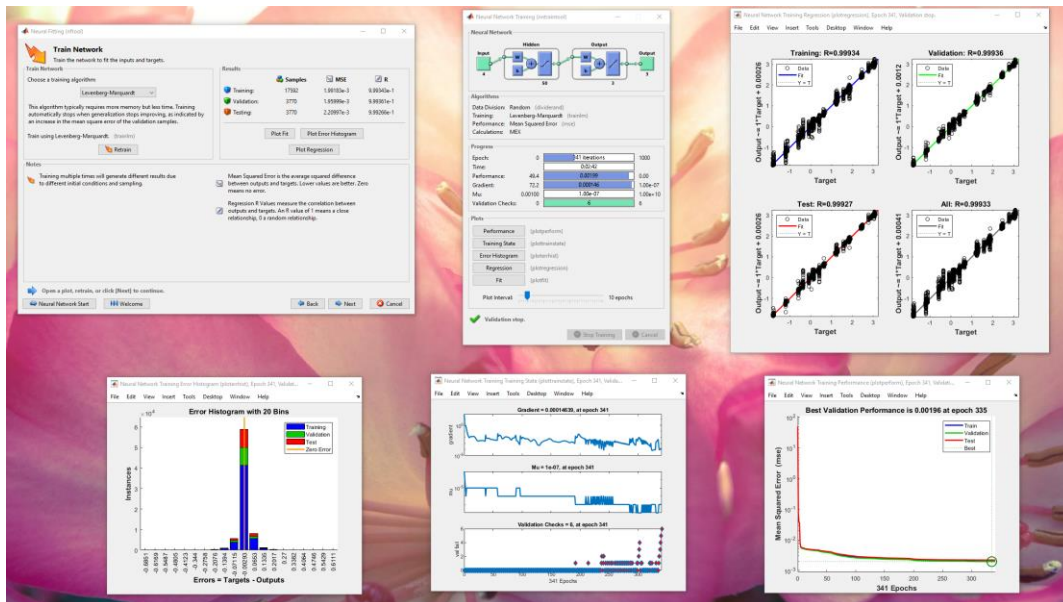


Figure 7.33: Training statistics for a 50 hidden layer ANN using four variable input for pared dataset within  $\pm 60$  degrees AoA.

Both the resulting raw fit point cloud and PDoA tag location averaged fit is shown in Figure 7.34. The location averaged data includes black connecting lines between the ground truth and model fit positions, whose thickness is proportional to the Euclidean error magnitude. These error lines are difficult to see in Figure 7.34 because the resulting errors are so small. The fit is excellent, but this may just mean that the ANN has been overfit to the data. Let us look at other mathematical models before returning to the question of ANN overfitting in the next section.

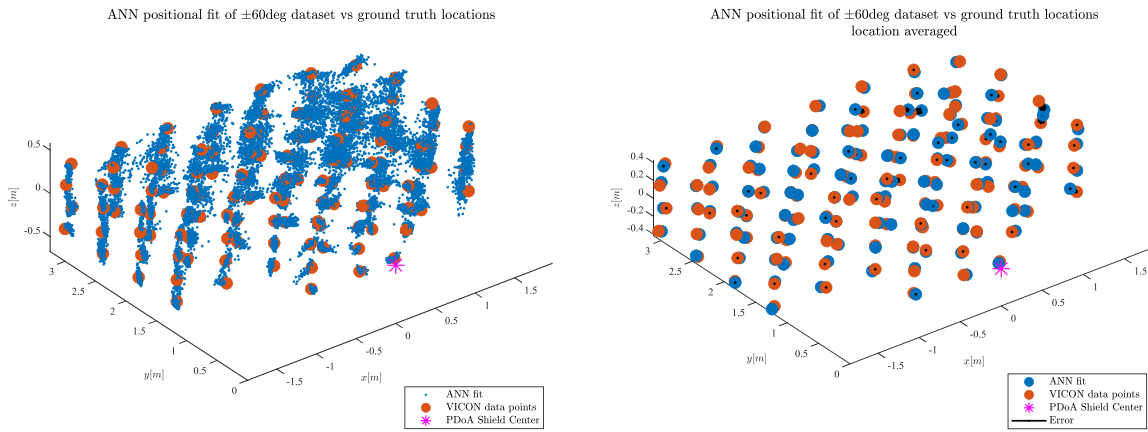


Figure 7.34: ANN performance trained on the  $\pm 60$  degrees dataset. Left: Raw point fits. Right: Location averaged fits.

The second and third models considered use less of a black box approach than the ANN and instead rely on the assumptions in equations (7.7) and (7.8). These relationships each have two inputs and one output, so they can be visualized in 3D plots. Two types of surface fits were tested for each of the horizontal and vertical AoA functional relationships. The first was simply a planar fit, also known as a first-degree polynomial in two variables. Equation (7.10) shows the relationship between the output AoA,  $f(x, y)$ , and the input PDoA variables,  $x$  and  $y$ , where all coefficients are constants.

$$f(x, y) = c_{00} + c_{10}x + c_{01}y \quad (7.10)$$

Once we have applied two such planar models to solve for each horizontal and vertical AoA, we convert the resulting angles and average UWB distance into cartesian coordinates using a reverse spherical coordinate transformation. The planar surface fits are shown visualized in Figure 7.35.

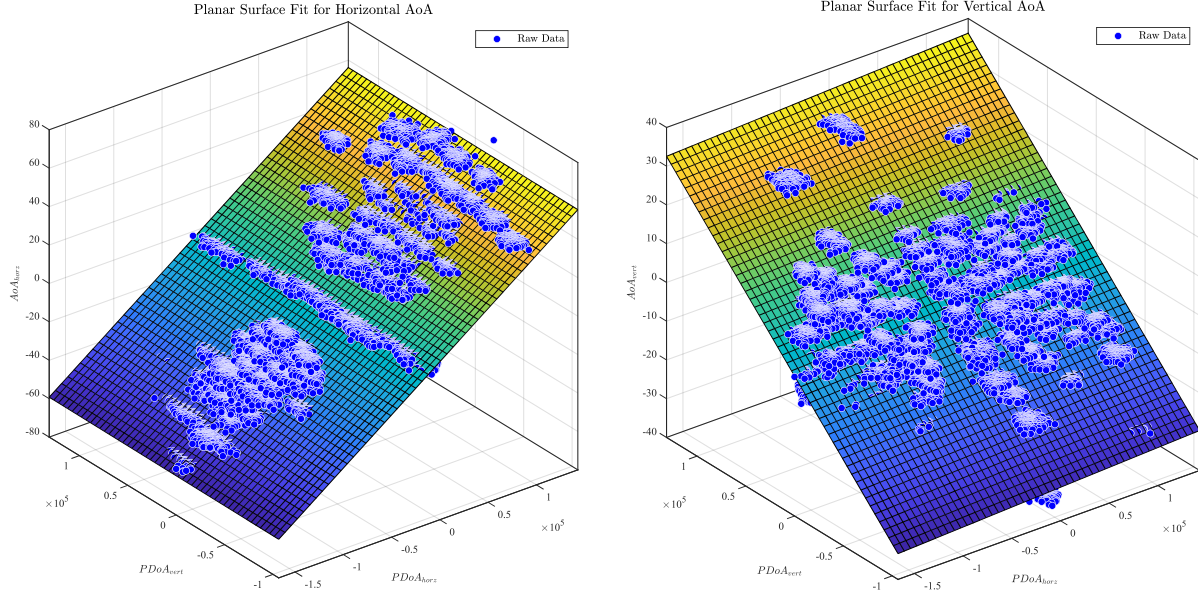


Figure 7.35: Planar surface fits for the two PDa input, single AoA output variable relationship assumption.

The resulting point cloud and location averaged fits are shown in Figure 7.36 where the error bars can now clearly be seen on the X-positive side of the figure. Overall, the fit looks coherent as can be expected. However, if the functions were truly linear, then this problem would not be a difficult one to solve. We know from the plots in the previous section that the raw data has some curve to it. Let us next try a mathematical model that incorporates curves.

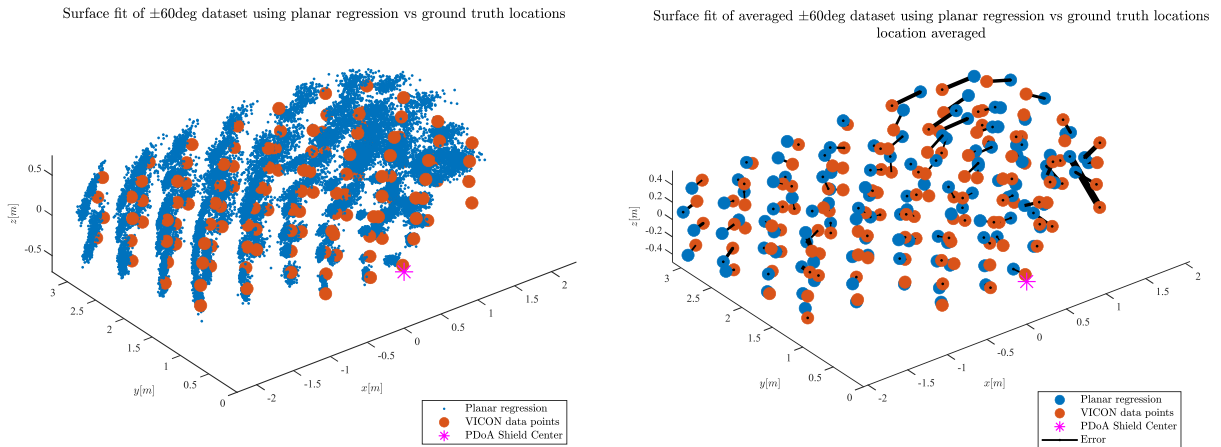


Figure 7.36: Planar performance trained on the  $\pm 60$  degrees dataset. Left: Raw point fits. Right: Location averaged fits.

The second surface fit explored was a third-degree polynomial in two variables, referred to here as a poly33 fit, to use MATLAB's terminology [138]. It may be more properly known as a bivariate cubic polynomial, but at any rate, equation (7.11) shows the relationship between

the output AoA,  $f(x, y)$ , and the input PDoA variables,  $x$  and  $y$ , where all coefficients are constants.

$$f(x, y) = c_{00} + c_{10}x + c_{01}y + c_{20}x^2 + c_{11}xy + c_{02}y^2 + c_{30}x^3 + c_{21}x^2y + c_{12}xy^2 + c_{03}y^3 \quad (7.11)$$

Intuitively, the plots in Figure 7.23 look as if they could be modeled by cubic polynomials. We have generalized that notion into 3D by adding a second variable to a one-dimensional cubic model. The poly33 surface fits are shown in Figure 7.37 in 3D plots. We can see that these surface fits do not have a large amount of curvature to them, but they still capture the nonlinear essence of the data better than the planar surface fits.

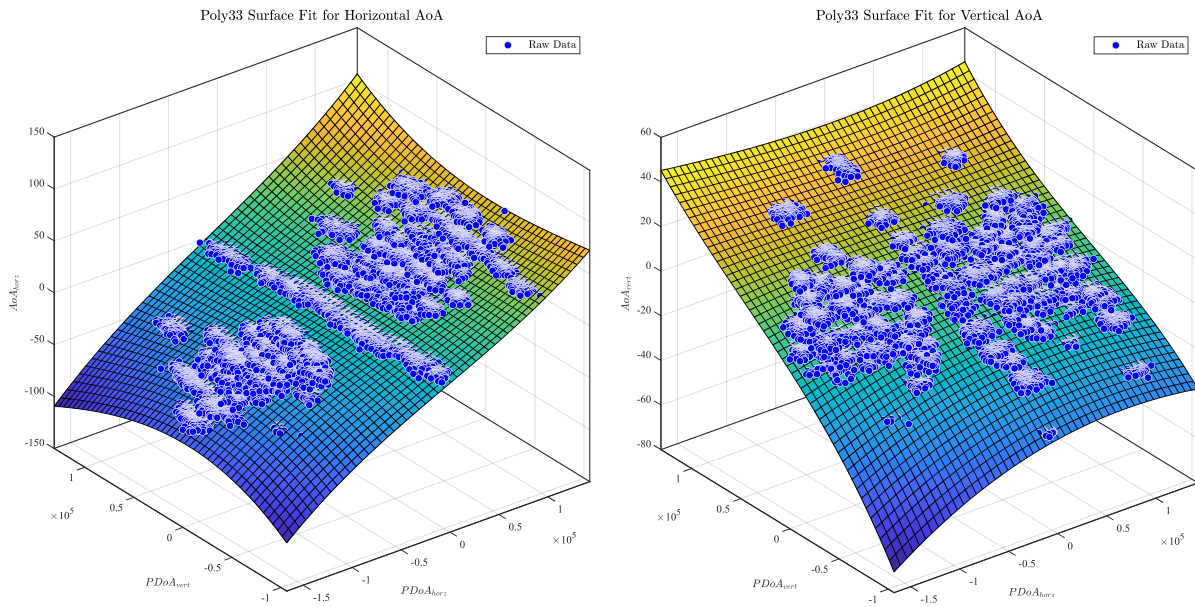


Figure 7.37: Poly33 surface fits for the two PDoA input, single AoA output variable relationship assumption.

The result of applying these mathematical models for calculating PDoA tag positions is shown in Figure 7.38, just as the previous results were presented. From the error bars, it can be observed, in terms of accuracy, that this model is somewhere in the middle between the ANN and planar models. There is one location with a large error, but otherwise Euclidean error looks to be small.

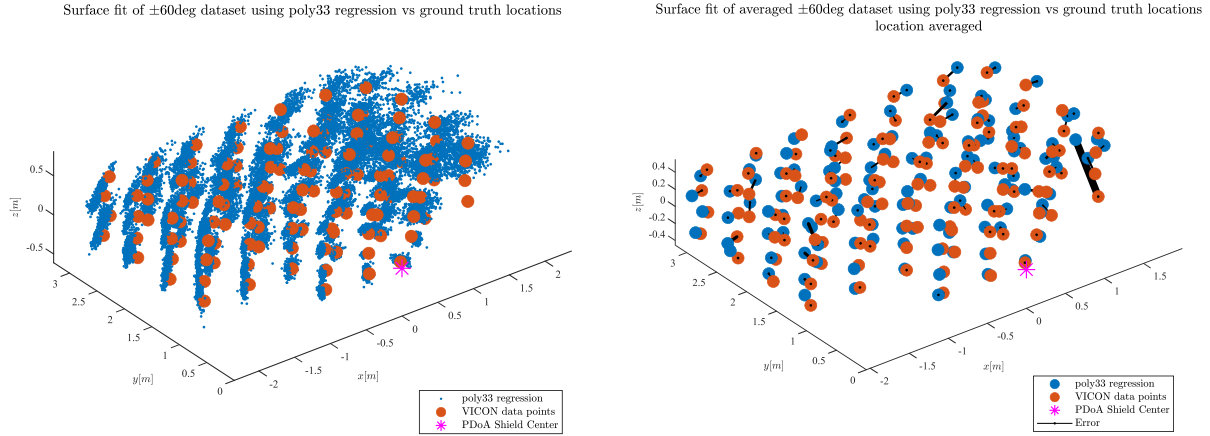


Figure 7.38: poly33 performance trained on the  $\pm 60$  degrees dataset. Left: Raw point fits. Right: Location averaged fits.

Now that we have qualitatively looked at the results of our modeling, let us investigate the Euclidean error distributions that arise from them. Figure 7.39 shows the error histograms of the three models overlaid on each other. The poly33 and planar surface fits have errors on the order of previous UWB results, but the ANN has a suspiciously low error of only 3.7cm. This is concerning, because just comparing ground truth radial distance to average UWB distance from the PDoA shield yields approximately 2-3x this amount of error. It is becoming apparent that this ANN has probably been overfit to the set of training data, as this low error is unrealistic.

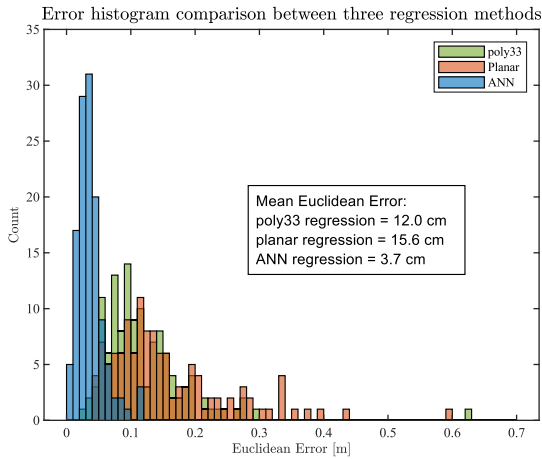


Figure 7.39: Histogram of Euclidean error to compare the accuracies of each of the three models investigated.

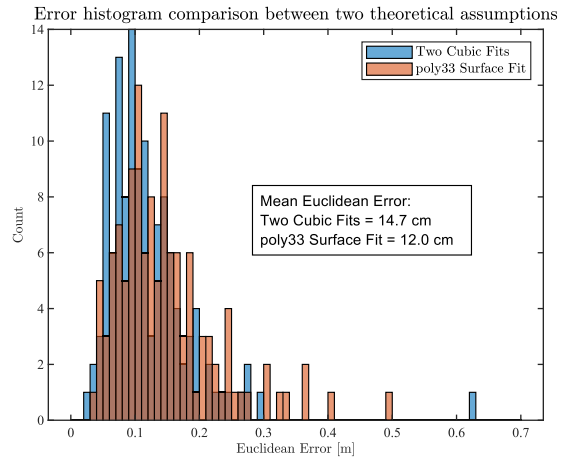


Figure 7.40: Comparison histogram of Euclidean error to validate the assumption of coupled PDoA nodes in 3D.

For a final note on the validity of our assumption that each of the PDoA nodes cannot be treated independently for modeling, Figure 7.40 shows the results of using two independent cubic polynomials with an assumption that AoA is only a function of the PDoA for the node in



the same plane. We can see that the results are not at all bad, but they are not as good as the poly33 fit. Clearly, the 2D plots in the previous section have some thickness and curvature to them that can be better captured by adding another independent variable to the models. Let us now turn our attention to model validation to see how these models perform on datasets other than the one that they were trained on.

## Model Validation

Any arbitrary functional relationship for a given dataset can be created using an ANN or any other suitably detailed mathematical model. This is not the goal of the current work, rather, we would like to create a model that operates well on many datasets sampled from the same of similar populations. To this end, we should test our models on different datasets to ensure that they are robust and consistent. To do this, the experiment was run again in two environments. The first environment was in the ECR of Figure 7.13 at three additional orientations, and the second environment was in the tent MoCap arena as shown earlier in Figure 5.6.

The first model validation results are shown in Figure 7.41 and their corresponding Euclidean error histograms in Figure 7.42. This setup used the PDoA shield oriented in the same pose as the original data was collected in. One small variation was that the PDoA tag was moved through a less ordered set of points than before to ensure that the models were not just overfit to specific locations of the PDoA tag. We can see from the Euclidean error metric that the two surface fits performed with near identical accuracy on this new dataset as the original dataset. As we are utilizing the assumption that the Euclidean error is sampled from a Gaussian distribution, their mean variations of less than a centimeter are within the standard error.

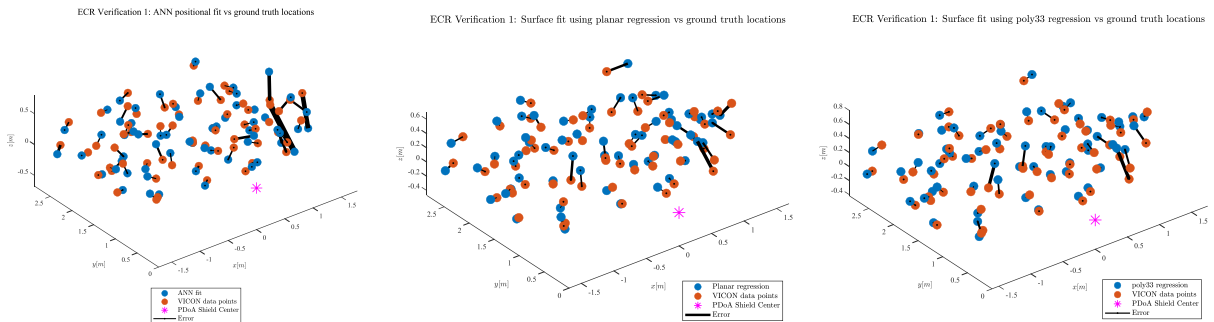


Figure 7.41: First set of model validations in the ECR. From left to right: ANN, Planar, poly33. Plots show model fit locations vs ground truth locations of the PDoA tag. Thicknesses of connecting error lines are proportional to error magnitude.



The ANN performance, on the other hand, has dramatically decreased to a larger mean error than we have observed in any of the models so far. However, less than 20cm of error is still on the same order of magnitude as the other models and not at all poor performance, so we cannot yet confirm overfitting to the training data.

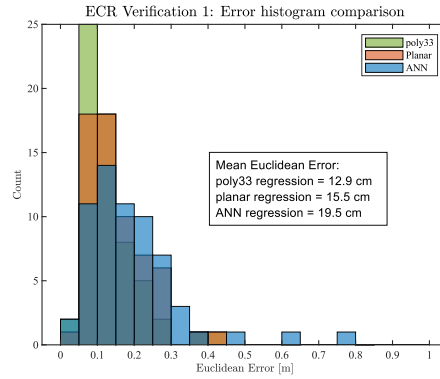


Figure 7.42: Comparison histogram of Euclidean errors from the first set of model validation runs in the ECR.

The second experimental scenario in the ECR was to swivel the PDoA shield around by 180 degrees and rerun the experiment on the other side of the MoCap arena. This area was the same size and shape as the original, as the MoCap space has symmetry about the Vicon X-axis. Results for this study are shown in Figure 7.43 and Figure 7.44. It can be observed that the ANN had a couple of major outliers, but overall, the models performed well in this new orientation.

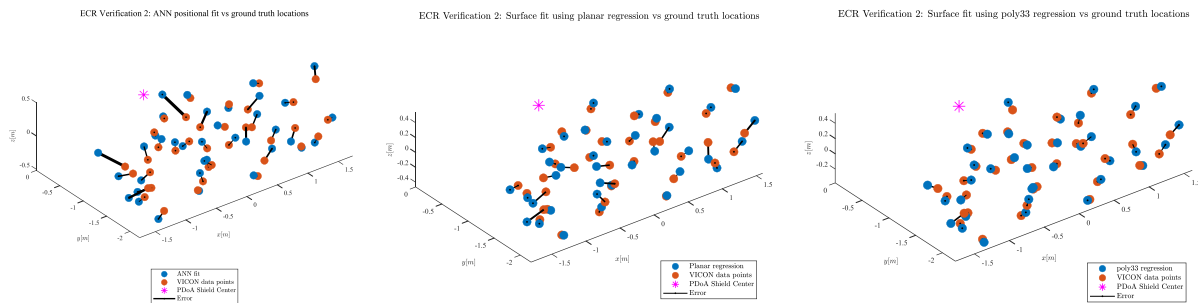


Figure 7.43: Second set of model validations in the ECR. From left to right: ANN, Planar, poly33. Plots show model fit locations vs ground truth locations of the PDoA tag. Thicknesses of connecting error lines are proportional to error magnitude.

In fact, the error histograms show that the models performed better in this setup, across the board, by a few centimeters of error each, compared to the last setup. The reason for this is not confirmed, but it may be the case that the PDoA shield rotation was simply clocked closer to the original dataset this time. The PDoA shield setup is a manual one that relies on operator

judgement to visually lineup the PDoA and Vicon frames, so non-negligible errors would not be surprising to encounter because of small misalignments.

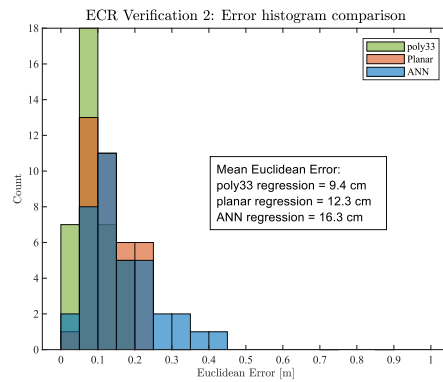


Figure 7.44: Comparison histogram of Euclidean errors from the second set of model validation runs in the ECR.

The third validation test performed in the ECR was to rotate the PDoA shield negative 90 degrees from the training experiment orientation. While still in the same lab space, this represented the greatest deviation from the original experiment, because the MoCap area used differed from the previous two setups in usable area and distance to the edge of the arena. Figure 7.45 and Figure 7.46 show the results of this test and indicate that error was greatest in this case for the surface fits, while the ANN performed the best in this trial.

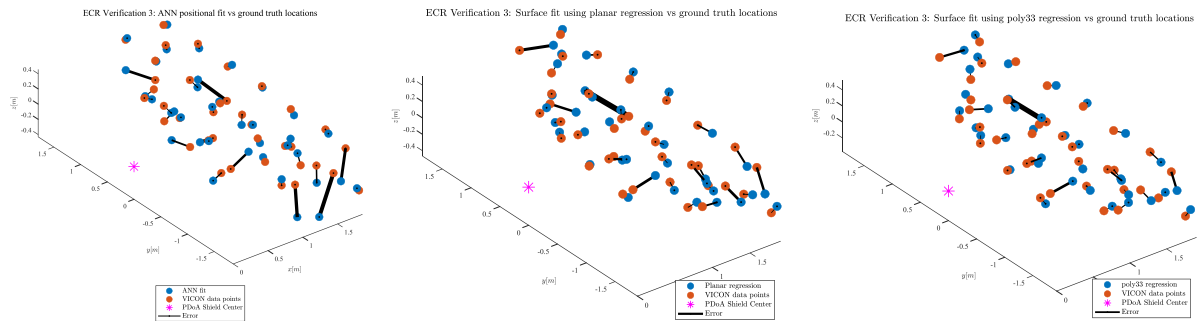


Figure 7.45: Third set of model validations in the ECR. From left to right: ANN, Planar, poly33. Plots show model fit locations vs ground truth locations of the PDoA tag. Thicknesses of connecting error lines are proportional to error magnitude.

Once again, operator setup may have played a role in the accuracy of the tests. The ANN performance cannot be readily explained, as is typically the case with such complex models. From the data collected, we can say that the PDoA shield is able to locate the PDoA tag in 3D to an accuracy of less than 20cm of Euclidean error, at least in the same environment that model training dataset was sampled from.

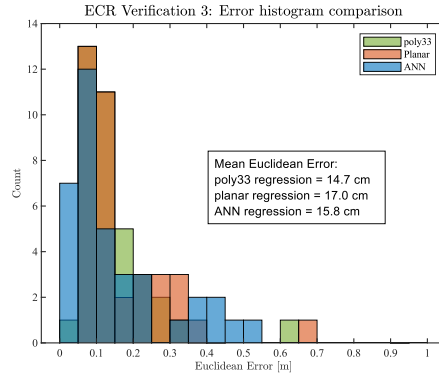


Figure 7.46: Comparison histogram of Euclidean errors from the third set of model validation runs in the ECR.

The last model validation to discuss is on the results of rerunning the original experiment in the tent MoCap arena. Each of the previous three model validations were performed in the same environmental space as the training dataset was sampled from. Radio transmission characteristics are specific to an environment based on many variables. The tent has a concrete floor as opposed to the carpeted floor of the ECR. The size of the tent space is also much larger than the ECR and any radio reflections will take that much longer to propagate from the walls and ceiling. The Vicon camera tripods are also taller and wider in the tent to accommodate a larger MoCap area than those used in the ECR. In summary, the tent is a much different radio environment than the lab room, although they are both uncluttered with LOS between the PDoA shield and tag.

The results of this final validation are shown in Figure 7.47 and Figure 7.48. At first glance of the ANN graph, it appears that there must have been a coding error in MATLAB due to the very large errors shown. However, after looking closely at the plot, about half of the points do show good accuracy. It appears that the ANN has overfit at least somewhat. It should be noted that the largest Y-coordinate tick in these plots is 5m, while the original dataset collection only occurred out to a maximum of 3m. Taking another look at the ANN plot, it appears that the model performs very well up to 3m, before accuracy rapidly decreases for the furthest row of points where the ANN maps the data to large negative Z-values. The takeaway from this analysis is that the ANN only performs well within the domain that it has been trained in and cannot be generalized beyond this. An analogy would be the 2D case of fitting a high-degree

polynomial to a set of data, whereby the curve may pass through each point perfectly, but then shoot off to positive or negative infinity on either ends of the dataset limit.

The visualized results for the surface fits look much better in Figure 7.47, although there is still a fair amount of error seen in the upper right quadrant of the area plotted. It may be that the PDoA hardware calibration was not perfect from the start and resulted in asymmetric data about the PDoA X-axis. Most of the plots in the modeling section did have points near this area of the dataset with large errors. This experiment was also run one year after the others due to the COVID-19 pandemic, so it is possible that the hardware was jostled enough during that time to change its calibration characteristics. The same PDoA tag was used throughout all experiments to maintain hardware consistency, as different tags will most likely have different error characteristics.

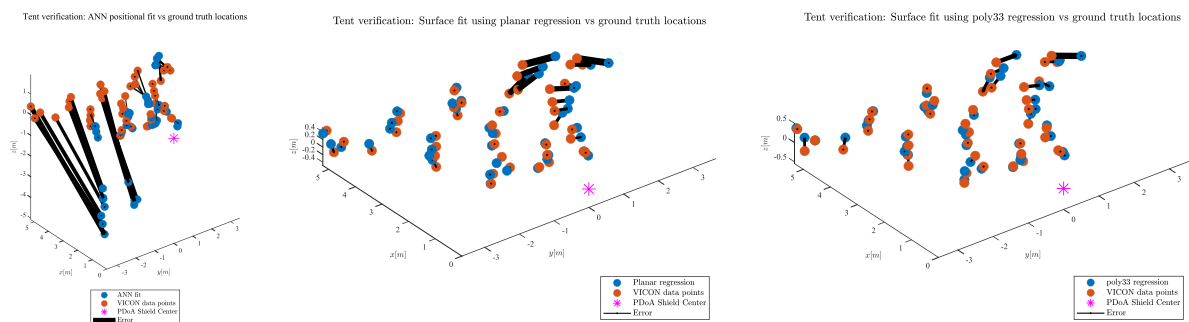


Figure 7.47: Model validations in the tent. From left to right: ANN, Planar, poly33. Plots show model fit locations vs ground truth locations of the PDoA tag. Thicknesses of connecting error lines are proportional to error magnitude. It is observed that the ANN model is grossly wrong at large radial distances from the PDoA shield. This confirms overfitting to the training data.

The Euclidean error histograms reveal performance that is slightly degraded, but on par with the other validation tests for the surface fits. The ANN errors shown are for the points at a small radial distance from the PDoA shield. The rest of the locations are outside of the range of the histogram, as can be seen in the mean error value of almost two meters! It appears from the data that the poly33 surface fit is the most accurate and robust model across multiple environments with a mean error of less than 20cm in all tests. As a conclusion to the model validations, 3D position using only one tag and one anchor is feasible using UWB with PDoA. There is more work to be done in hardware and model optimization to ensure repeatability and robust accuracy in many more environments, but the experimentation in this chapter shows that a real-world solution should be possible.

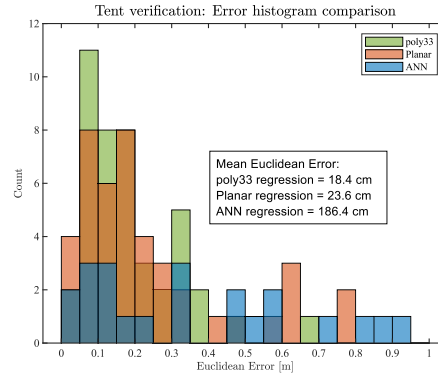


Figure 7.48: Comparison histogram of Euclidean errors from the model validation run in the tent. The full ANN error distribution extends twice the scale of this plot because the model errors are so large.

## Discussion

### PDoA Error in NLOS Conditions

The geometry of the PDoA antenna array is what allows it to measure AoA from an arriving signal. While it is true that UWB is resilient to multipath propagation due to the short time duration of its impulses, reflected paths will still fundamentally result in the calculation of a wrong AoA. PDoA-based approaches to localization require line-of-sight for high accuracy due to the directionality inherent in their design [53]. If there is radio NLOS for the UWB frequency range, then PDoA will not operate with accuracy as the wrong spherical angles will result in miscalculation of cartesian coordinates. Additionally, AoA localization deteriorates as transmitter-receiver distance increases due to small errors in angle estimation amplifying into large errors in position calculation [53]. The error/distance relationship was not studied explicitly here, but it should be explored in future research.

### Improvements

This research was carried out with beta-level development hardware and software. While capable for investigation purposes, the equipment has many areas of possible improvement. The following subsections will discuss details of these possible improvements to operation.

#### Real-Time Capability

The main barrier to implementing the PDoA shield as a real-time positioning system was the lack of capability for each node not to not interfere with the other. For real-time implementation, a frame sequencer is required to allow each PDoA node to switch back and

forth and share airtime. Although simple in theory, such a change to the driver code was out of the scope of this research. There were several attempts made at cleverly sidestepping this issue, including utilizing a separate tag to communicate exclusively with each node as well as rapidly switching between the nodes in an attempt at a 50% duty ratio for each. Multiple tags did not solve the issue and it was determined that continuously running each node simultaneously on the same channel with the same signals jammed the channel. Neither was rapid switching an option, as it takes several seconds for the each PDoA node to switch between stop and start modes. Thus, the decision was made to proceed with static position capabilities designed in a way to facilitate frequent movement of the tag for data collection. Although the capability of the PDoA shield discussed is not real-time, it still provides use for situations in which static conditions can be guaranteed or at least assumed for most of the time. Future implementations should be designed to allow multiple PDoA measurements in real-time from multiple antenna pairs.

#### *Hardware Redundancy*

Currently, each of the four antennas on the PDoA shield is connected to its own DW1000 chip and the PDoA post-processing is done on a microprocessor. Decawave is releasing a new chip, the DW3000, that brings PDoA processing onboard and eliminates the need for multiple UWB chips, thus reducing power consumption. Exact details are not known on the capabilities of this device, but the author suspects that it will integrate all the aspects of the beta PDoA kit into a single chip. That is, each DW3000 will probably support one pair of differential antennas for PDoA localization in the plane. Thus, 3D localization will still require multiple chips, but a lesser number of such than the current situation.

#### *60-Degree Limitation*

The limitation on the vertical node returning good data past  $\pm 60$  degrees of horizontal angle is intrinsic to the PDoA shield design, but more antennas may be added to increase the effective angular range of the PDoA shield. Another pair of antennas added between the horizontal and vertical at 45deg should increase range all the way to  $\pm 90$  degrees in both horizontal and vertical axes, which is currently the angular limit of any PDoA node. This would easily increase the angular range to the figures cited in previous papers and verified during these experiments



for a single axis. This works, because the horizontal PDoA is clearly not restricted to  $\pm 60$ -degree operation in the plane, although the data does become more nonlinear above this limit. Further testing and design are required to optimize operational limits, but it should be possible to reach  $\pm 90$  degrees operation with additional antenna pairs.

### *90-Degree Limitation*

The  $\pm 90$ -degree measurement limitation is entirely due to the PCB antennas only working in front of their fiberglass backing. This limitation could be eliminated by designing free-standing 360-degree antennas for the PDoA application that have constant omnidirectional gain. Logic would have to be included to uniquely determine directionality of the impinging signal, as the PDoA signal domain is  $[-180, 180]$  degrees already and this is achieved in one hemisphere around the PDoA shield. If each pair of antennas could measure PDoA in a full plane, then four antennas could cover the entire 3D space. This could work with any number of antenna pairs, although the convenience of data visualization would be lost with more than 2 pairs. In this case, the regressions would have to be generalized into more than three dimensions and visualized slice-by-slice, but the methods used in this chapter have no obvious barrier to further dimensional generalization.

### *Incomplete Dataset*

There is a valid argument to be made that these experiments have collected data only within a subset of the full range of the PDoA shield. This was due to the physical constraints of the Vicon hardware available, namely the limited height of the tripods, along with the limited space available in the ECR. Nonetheless, the working assumption is of symmetry in the system. One would expect that should the horizontal and vertical PDoA nodes be reversed, with greater range in the vertical and limited range in the horizontal, that the results would be identical but reversed between the nodes. This experiment could be carried out artificially by rotating the PDoA shield 90 degrees about the PDoA shield X-axis and rerunning the experiment.

Figure 7.49 shows comparisons of the assumed operating region using a 60-degree right cone, versus the operating region tested in the experiments as visualized by the PDoA tag location data points. The angular range is a function of the space available to increase distance for

measurements and greater distances have not been verified to have the same  $\pm 60$ -degree range. A different experimental setup could alleviate this by collecting more data within the assumed cone of operation to verify consistency with these results or to yield new results. One possible way to do this with the current Vicon MoCap system would be to mount the PDoA shield on the ceiling to better test the angular range, however this would mean reduced resolution in distance measurements as limited by the height of the system. The ideal experimental setup to test this valid operating region assumption would be the PDoA shield mounted in the ceiling of a tall tipi with a drone flying around an attached PDoA tag.

#### Assumed Valid Modelling Region

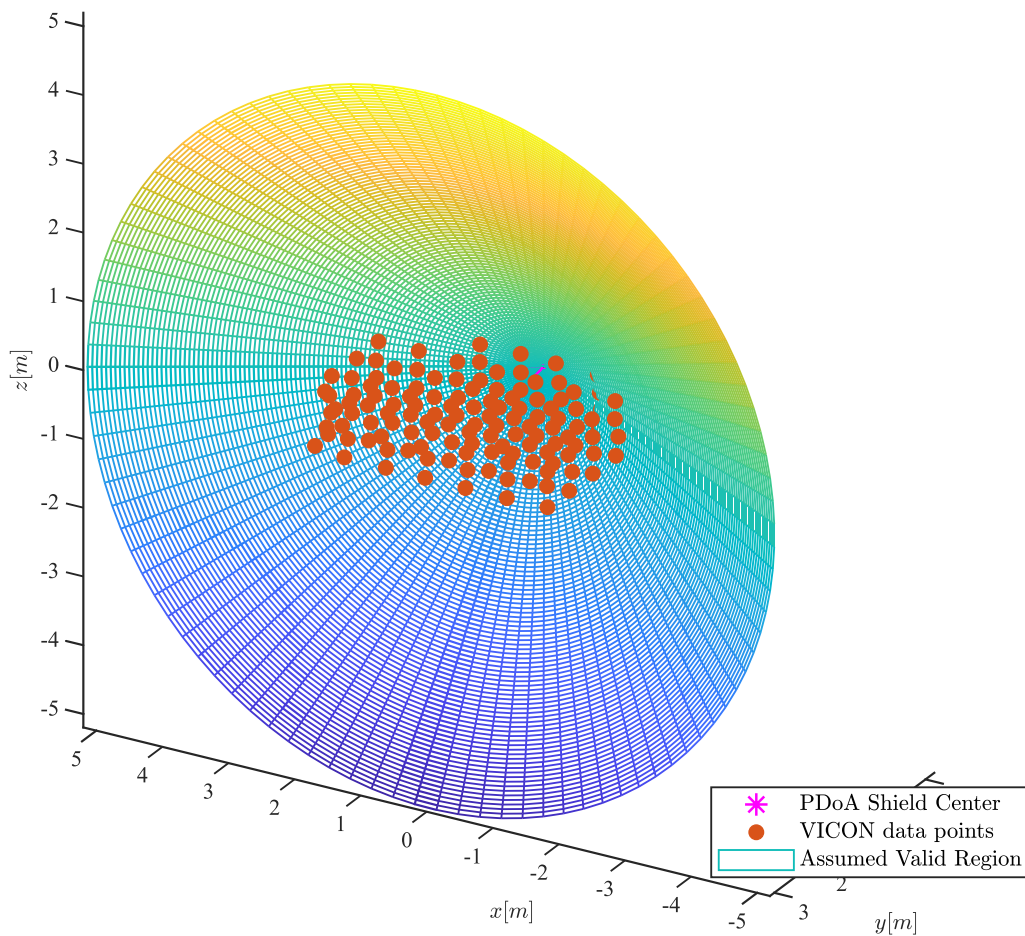


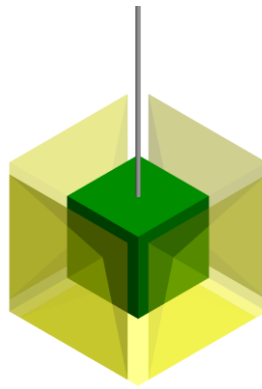
Figure 7.49: The 60-degree right cone is the region in which we have implicitly assumed that the models are valid in. The data points are the region that the PDoA system has been tested in.

## Applications

Possible applications of a single-anchor 3D tracking UWB system are endless. They include all the previously discussed applications for robot localization and many more. Such a system would allow much simpler setup than a multilateration anchor mesh and reduce the barriers to installation in environments such as homes, grocery stores, and any number of environments with ceiling mounting space. Let us briefly discuss possible solutions to circumvent angular range issues before diving into one specific application.

### *PDoA Shield Mechanisms*

The PDoA shield as it has been treated in this chapter only works within a right cone of 60 degrees. This geometry would work mounted to a ceiling, given ample height to track either ground motion or elevations of interest. Multiple PDoA shields could even be combined into a single unit to fully track 3D in a room, as shown in Figure 7.50, for example in a drone arena.



*Figure 7.50: Conceptual diagram of a stationary, hanging, centralized drone arena tracking module.*

Other possibilities for 3D tracking using the current PDoA shield incorporate active axes into mechanisms with rotary encoders. The Trimble S3 total station is an example of a horizontal angle axis that could rotate with feedback from the PDoA shield sensor to track a PDoA tag 360 degrees around it. Indeed, the PDoA shield could be mounted on an additional trunnion axis too, as the laser is in the total station, and have full 3D tracking abilities without the current restriction of half-hemisphere operation.

### *Initialization of Standard UWB Anchors*

This application leverages the single-anchor UWB PDoA shield to accurately localize traditional multilateration UWB anchors. A laser EDM and camera are paired with the mechanisms discussed in the previous section and a three-step process is utilized to first roughly locate an anchor with UWB PDoA, then accurately locate the antenna using machine vision, and finally measure the anchor distance with the laser. The UWB anchors are then initialized to provide for real-time localization of a mobile robot. This method would greatly increase the feasibility of autonomously deployment and initialization of an UWB anchor mesh network. Possible methods of deployment include, but are not limited to, mounting anchors on UAVs or UGVs for active deployment or dropping off anchors on tripods in the environment for passive deployment.

### *Conclusion*

In this chapter, we have discussed the feasibility of utilizing single anchor UWB PDoA for 3D localization of tags. Results indicate that, given the proper hardware setup and mathematical model to transform raw data into cartesian position, it is possible to accurately locate within a Euclidean ball of radius of less than 20cm at distances up to 5m, within a spherical angular range of  $\pm 60$  degrees in each axis. There is still much more research to be conducted further on this problem and some of the various avenues of inquiry have already been highlighted throughout the chapter.

## CHAPTER 8: Future Research

There are countless opportunities for applied research utilizing UWB and other technologies discussed in this thesis. Already, some small and potentially large problems have been highlighted within the text. This will be a discussion of additional high-level opportunities for further research within the field.

An obvious barrier to the use of UWB multilateration for robot localization is the need to setup a mesh of anchor antennas in the desired environment. UWB PDoA was one possible solution to this issue, but there are others. Ideally, autonomous robotics will one day require no intervention from human operators. It would be desirable to achieve a means of autonomously deploying an UWB anchor network or somehow integrating this into the robots themselves. One possible solution would be to mount UWB anchors on small mobile ground robots and drones. These robots could live on a larger carrier vehicle until a destination environment was reached and the small robots spread out to provide an anchor mesh. Ground robots could use tripod mechanisms to make themselves tall for LOS and drones could attach to the edges and faces of buildings. Each of these small robots would be performing their own localization and an optical EDM could refine their position estimates much like a total station operates. Better still would be a system in which the UWB anchor robots did not have to remain static. Robots for other uses could carry UWB anchors and create a dynamic mesh for localization via multilateration as has been explored in [139]. There could arise difficulties in the geometry of solving the multilateration algorithm in this case. Another extension to make UWB more useful in this scenario is an anchor hopping algorithm like cellphones use to switch between UWB cells as has been explored in [140].

Aside from autonomous fleets, robots also have the potential to revolutionize the construction industry. Already, farm equipment and asphalt laying machines are using GNSS for localization, but UWB has the potential to improve localization in cluttered urban environments without direct LOS to the sky. UWB could augment or replace GNSS in these environments and provide a means to automate site leveling and foundation digging. Indeed, there is a great deal of research to be done on autonomous robotics for earth moving. Even now, teleoperated

construction robotics are not yet entirely feasible due to the detailed feedback that operators require from feeling the forces on the vehicle that are near impossible to recreate through a remote-controlled interface. These problems and many others make autonomous construction robots in many ways more difficult to solve than driverless cars constrained to roadways.

This thesis has mostly discussed ground robotics, where pose estimation for a rigid body requires only three variables: two position coordinates and an angular heading. UWB is generalizable to 3D pose estimation, but the system and techniques must be modified slightly. 3D position is the easiest to achieve and has been already in practice. Accurate elevation estimation requires the same degree of UWB anchor spread in the Z-axis as is seen in the plane. This is easily achievable for drones in a small lab space with eight UWB anchors in the corners of the room, but more difficult at the scales required for large drones. The reasoning for this required spread is the same reason that GNSS is less accurate in the vertical dimension [141]. 3D orientation using two or more UWB tags has the same requirements as heading determination discussed in previous chapters. The further apart the tags are from each other, the more accurate orientation estimates will be. As was seen with the J8, this requires a large vehicle and typical drones do not have this amount of dimension to spread UWB tags across. A very large drone would be required and that itself would present technical challenges. Otherwise, the mathematical treatment is the same as that for heading determination and generalizing orientation estimation to 3D would not be difficult.

The last area of research to be discussed is ground robotics in the presence of elevation changes. Based on the success of robot localization in both 2D and 3D, ground elevation changes would seem not to be an issue, but such an environment is harder for robots than either 2D or 3D. We will refer to a ground robot in an environment with steep changes in elevation as operating in a 2.5D space. Ground robotics have the benefit of only three dimensions to consider for localization and movement. Drones need to consider all six dimensions, but they have the luxury of actuating in free space and idealizing the ground as 2D, even if there is hilly terrain. For instance, a ground following drone could maintain constant height from the ground while watching out for trees and buildings and not have to worry too much about a steep incline due to the elevation buffer that gives them to react and adjust



height. Ground robotics in hilly terrain have neither of these luxuries. They are affected by steep inclines because this causes their lidar to see no obstacles on the uphill side and a solid wall on the downhill side. After cresting the hill, the situation reverses and there will need to be enough logic present to let the robot know that it can continue driving safely ahead without clearing the obstacle map behind it. 2.5D autonomous robotics have the requirement to track full 3D pose information without the luxury of movement in free space. These factors make this class of problem difficult to solve and there is still much work to be done in making robots more intelligent in this area for obstacle detection and robot localization.

These are just a few of the ongoing areas of research for UWB and robot localization. Of course, there are too many opportunities to list here, but the aspiring researcher or roboticist should have no issues finding problems to tackle in this exciting field.

## CHAPTER 9: UWB Commercialization Potential

Ultra-wideband (UWB) localization utilizes a 500MHz or greater radio spectrum to create very short pulses in time [142]. These short pulses are ideal for discerning multi-path signal reflections [54]. This property makes UWB superior to narrowband localization methods using Wi-Fi or Bluetooth. UWB antenna meshes can locate to within 10cm positional accuracy using trilateration [143]. Additionally, the technology is resilient to changes in the operating environment, which it makes it ideal for real-world deployment [144]. Already, it has found applications in warehouse asset tracking and interactive museum exhibits [145] [146]. Indeed, it is more accurate than GPS and has the advantage of operating in all spaces that have GPS denial or degradation. Three possible use cases will be discussed in different environments and the market opportunities and challenges for each case.

The first use case is in outdoor applications. There are many situations where GPS alone does not suffice. Downtown areas are packed with skyscrapers that make GPS unreliable and noisy [147]. UWB would be an excellent substitute. High-accuracy construction asset tracking, autonomous robotics, and mapping are enabled by UWB. End users would be laborers that could setup a mobile system and then connect all their instruments to it. It would enable movement tracking and ensure that workers stayed out of danger. Such a system would have to be rugged and waterproof, but otherwise would not have to be consumer friendly. The B2B price point would support selling specialized pieces of hardware and custom software. Revenue in this application would come from hardware modules and robotics integrations. Competitors in this industry like Caterpillar [148] and Trimble [149] already offer some solutions. The most feasible entry option in this use case would be a well-funded startup to work on prototyping and algorithms with the goal of being acquired by one of these established companies.

The other common domain that UWB finds use is the indoor environment. Possibilities range from concerts to conventions to common shopping scenarios. Consumers want to have a map on their phones with a blue dot to locate them inside just as easily as outside. The benefit of this application would be much more apparent to venue owners and operators. Data would be the ultimate product in this case. Much like website metrics, there would be a wealth of data

enabled to show where consumers navigated, gathered, and lingered. This valuable data would allow layout optimizations to maximize marketing and sales. As a result, the business model for UWB here would revolve around big data and its analysis. The largest barrier to such an application is currently the lack of UWB in smart phones, but that is coming [150]. Apple now has an UWB chip called the U1 in their phones [151] and Decawave, the company featured in this work, was recently acquired by Qorvo [152], a supplier to Apple [153]. The week that this thesis was published, Apple announced and released its new AirTag, a UWB and Bluetooth accessory for locating lost items with an iPhone [154]. Samsung and other manufacturers are sure to follow Apple's lead. These large competitors will dominate, but hopefully provide an API to interface with their UWB modules. The smartest entry into this market would be to start making data analysis software now in anticipation of interfacing in the future. Another revenue generating venture would be a service business that installs and calibrates UWB antenna systems in all the areas of interest, as we have seen the difficulty of anchor mesh setup.

The final use case to be discussed is that of the underground. This environment includes caves, tunnels, and mines. There is currently an established UWB startup working on the problem of train control in tunnels [155]. The most vexing problem in these environments is how to calibrate the system in the first place. All the fixed UWB antennas must know their locations to provide capability of finding moving targets. A long underground passageway would require surveyors to move from a known location topside and find locations of all antennas in a time-consuming fashion. Once a system was setup, pedestrians, miners, spelunkers, or robots would have localization capabilities to support navigation and mapping. Miners could be tracked and easily located in case of collapse based on their last known location. Autonomous robots could patrol the underground taking measurements, delivering payloads, or otherwise replacing manual labor. In this case, customers would be mine operators who would not bat an eye at investing in these capabilities. The product would be a new application of robots that are currently used in warehouses but are lacking the infrastructure to operate in mines [156]. Ruggedness and simplicity of operation would be valued over an open interface. This is an area where one could have a competitive advantage bringing a new technology to market.

In summary, there are multiple commercial applications for UWB localization. Uses are coming to market right now and we can expect UWB to be ubiquitous soon. This is an excellent time to create a startup based on the technology before the market becomes completely saturated and dominated by a few established players.

## CHAPTER 10: Conclusion

In this thesis, UWB has been extensively studied from the practical perspective of robot localization. Several systems have been statistically characterized, in multiple different environments. It has been verified that multilateration UWB systems can provide accurate pose estimates to robots in spaces large enough to be practically useful and in GNSS-denied environments where there are limited options for any comparable system other than UWB. Additionally, other novel sensors for robot localization have been developed, including the Trimble SX-10 total station and the PDoA shield that is constructed of Decawave beta PDoA kits. Further avenues of research have been discussed with a focus on practical applications and problems in robot localization. The commercial potential of UWB has been discussed and the author believes that society is on the verge of seeing UWB become ubiquitous in our daily lives for much more than robot localization. For the interested reader, all data, source code, CAD models, and more from this thesis have been included in a GitHub repository referenced in the appendix. If you have read through until this point, thank you and good luck.

## REFERENCES

- [1] C. Dempsey, "Mapping Through the Ages: The History of Cartography," 29 January 2011. [Online]. Available: <https://www.gislounge.com/mapping-through-the-ages/>. [Accessed 08 April 2021].
- [2] J. B. Harley, "The Map and the Development of the History of Cartography," in *The History of Cartography*, vol. 1, Chicago, University of Chicago Press, 1987, pp. 1-2.
- [3] J. Evans, "History of astronomy," [Online]. Available: <https://www.britannica.com/science/astronomy/History-of-astronomy>. [Accessed 08 April 2021].
- [4] Canadian Space Agency, "A brief history of astronomy," 12 March 2020. [Online]. Available: <https://www.asc-csa.gc.ca/eng/astronomy/basics/brief-history-astronomy.asp>. [Accessed 08 April 2021].
- [5] D. Whitehouse, "'Oldest star chart' found," *BBC News World Edition*, 21 January 2003.
- [6] C. Thompson, "From Ptolemy to GPS, the Brief History of Maps," *Smithsonian Magazine*, July/August 2017.
- [7] National Geographic, "Navigation," National Geographic, [Online]. Available: <https://www.nationalgeographic.org/encyclopedia/navigation/>. [Accessed 10 April 2021].
- [8] A. J. Bratcher, "History of Navigation at Sea," 2008. [Online]. Available: <http://www.waterencyclopedia.com/Mi-Oc/Navigation-at-Sea-History-of.html>. [Accessed 13 April 2021].
- [9] S. Åkesson, J. Boström, M. Liedvogel and R. Muheim, "Animal navigation," in *Animal Movement Across Scales*, L. Hansson and S. Åkesson, Eds., Oxford University Press, 2014, pp. 151-178.
- [10] R. K. Zimmer-Faust, C. M. Finelli, N. D. Pentcheff and D. S. Wetthey, "Odor Plumes and Animal Navigation in Turbulent Water Flow: A Field Study," *The Biological Bulletin*, vol. 188, no. 2, April 1995.
- [11] J. L. Gould, "Animal navigation," *Current Biology*, vol. 14, no. 6, pp. R221-224, 2004.
- [12] Smithsonian, "Early Life on Earth – Animal Origins," Smithsonian, [Online]. Available: <https://naturalhistory.si.edu/education/teaching-resources/life-science/early-life-earth-animal-origins>. [Accessed 08 April 2021].
- [13] R. Siegwart and I. Nourbakhsh, "Mobile Robot Localization," in *Autonomous Mobile Robots*, 1st ed., The MIT Press, 2004, pp. 159-230.
- [14] S. Huang and G. Dissanayake, "Robot Localization: An Introduction," 15 August 2016. [Online]. Available: <https://onlinelibrary.wiley.com/doi/full/10.1002/047134608X.W8318>. [Accessed 08 April 2021].
- [15] D. Tang, "Localizatoion," 01 March 2017. [Online]. Available: <https://www.cpp.edu/~ftang/courses/CS521/notes/Localization.pdf>. [Accessed 11 April 2021].



- [16] ROS, "geometry\_msgs/Pose Message," 13 January 2021. [Online]. Available: [http://docs.ros.org/en/noetic/api/geometry\\_msgs/html/msg/Pose.html](http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/Pose.html). [Accessed 08 April 2021].
- [17] P. Goel, S. I. Roumeliotis and G. S. Sukhatme, "Robot Localization Using Relative and Absolute Position Estimates," University of Southern California, Los Angeles, CA.
- [18] W. Meeussen, "REP-105 | Coordinate Frames for Mobile Platforms," 27 October 2010. [Online]. Available: <https://www.ros.org/reps/rep-0105.html>. [Accessed 08 April 2021].
- [19] The Editors of Encyclopaedia Britannica, "Dead reckoning," 18 February 2005. [Online]. Available: <https://www.britannica.com/technology/dead-reckoning-navigation>. [Accessed 08 April 2021].
- [20] W. Seo and K.-R. Baek, "Indoor Dead Reckoning Localization Using Ultrasonic Anemometer with IMU," *Journal of Sensors*, vol. 2017, p. 12, 19 April 2017.
- [21] S. Campbell, N. O. Mahony, A. Carvalho, L. Krpalkova, D. Riordan and J. Walsh, "Where am I? Localization techniques for Mobile Robots," in *2020 6th International Conference on Mechatronics and Robotics Engineering*, Barcelona, Spain , 2020.
- [22] R. Burnett, "Understanding How Ultrasonic Sensors Work," 24 March 2020. [Online]. Available: <https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm>. [Accessed 11 April 2021].
- [23] J. Borenstein and Y. Koren, "Obstacle Avoidance with Ultrasonic Sensors," *IEEE Journal of Robotics and Automation*, vol. 4, no. 2, pp. 213-218, April 1988.
- [24] D. Vivet, P. Checchin and R. Chapuis, "Localization and Mapping Using Only a Rotating FMCW Radar Sensor," *Sensors*, vol. 13, no. 4, p. 4527–4552, 8 April 2013.
- [25] Delphi Automotive PLC, "Delphi's Industry-Leading Electronically Scanning Radar Brings Safety to One Million Vehicles," 21 August 2014. [Online]. Available: <https://ir.aptv.com/investors/press-releases/press-release-details/2014/Delphis-Industry-Leading-Electronically-Scanning-Radar-Brings-Safety-to-One-Million-Vehicles/default.aspx>. [Accessed 13 April 2021].
- [26] L. Stanislas and T. Peynot, "Characterisation of the Delphi Electronically Scanning Radar for Robotics Applications," in *Proceedings of the Australasian Conference on Robotics and Automation 2015*, 2015.
- [27] E. Ward and J. Folkesson, "Vehicle localization with low cost radar sensors," in *2016 IEEE Institute of Electrical and Electronics Engineers Intelligent Vehicles Symposium (IV)*, 2016.
- [28] R. P. Guan, B. Ristic, L. Wang, B. Moran and R. Evans, "Feature based moving robot localization using Doppler radar: Achievable accuracy," in *2017 20th International Conference on Information Fusion (Fusion)*, Xi'an, 2017.
- [29] R. Rouveure, C. Debain, R. Peuchot and J. Laneurit, "Robot Localization and Navigation with a Ground-Based Microwave Radar," in *2019 International Radar Conference (RADAR)*, Toulon, France, 2019.
- [30] A. S. Mohammed, A. Amamou, F. K. Ayevide, S. Kelouwani, K. Agbossou and N. Zioui, "The Perception System of Intelligent Ground Vehicles in All Weather Conditions: A Systematic Literature Review," *Sensors*, vol. 20, no. 22, p. 6532, 15 November 2020.

- [31] K. L. Ho and P. Newman, "Loop closure detection in SLAM by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740-749, 30 September 2006.
- [32] P. K. Panigrahi and S. K. Bisoy, "Localization strategies for autonomous mobile robots: A review," *Journal of King Saud University - Computer and Information Sciences*, 11 March 2021.
- [33] W. Zhi, J. Chu, J. Li and Y. Wang, "A Novel Attitude Determination System Aided by Polarization Sensor," *Sensors*, vol. 18, no. 1, 9 January 2018.
- [34] A. Trebi-Ollennu, T. Huntsberger, Y. Cheng, E. T. Baumgartner, B. Kennedy and P. Schenker, "Design and Analysis of a Sun Sensor for Planetary Rover Absolute Heading Detection," *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 17, no. 6, pp. 939-947, December 2001.
- [35] C. Boirum, "Improving Localization of Planetary Rovers with Absolute Bearing by Continuously Tracking the Sun," Carnegie Mellon University, Pittsburgh, Pennsylvania, 2015.
- [36] F. Cozman and E. Krotkov, "Robot Localization using a Computer Vision Sextant," Carnegie Mellon University, Pittsburgh, PA.
- [37] D. A. Sigel and D. Wettergreen, "Star Tracker Celestial Localization System for a Lunar Rover," in *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, 2007.
- [38] M. Venezia, "What is the Difference Between GNSS and GPS?," 16 December 2015. [Online]. Available: <https://www.semiconductorstore.com/blog/2015/What-is-the-Difference-Between-GNSS-and-GPS/1550/>. [Accessed 11 April 2021].
- [39] D. Hambling, "GPS chaos: How a \$30 box can jam your life," *NewScientist*, 2011.
- [40] C. Lopez, "Multipath," European Space Agency, 19 July 2011. [Online]. Available: [https://gssc.esa.int/navipedia/index.php/File:Multipath\\_Fig\\_1.png](https://gssc.esa.int/navipedia/index.php/File:Multipath_Fig_1.png). [Accessed 11 October 2020].
- [41] T. Hitchens, "SASC Wants Alternative GPS By 2023," *Breaking Defense*, 2020.
- [42] A. Bensky, *Wireless Positioning Technologies and Applications*, 2nd ed., Norwood, MA: Artech House, 2016.
- [43] M. Luccio, "The Return of Loran," 2020.
- [44] Continental Electronics Corporation, "eLORAN TRANSMISSION SYSTEMS," 2021. [Online]. Available: <https://contelec.com/eloran/>. [Accessed 11 April 2021].
- [45] Z. Sahinoglu, S. Gezici and I. Güvenc, "Ultra-wideband signals," in *Ultra-wideband Positioning Systems: Theoretical Limits, Ranging Algorithms, and Protocols*, Cambridge University Press, 2008, pp. 20-43.
- [46] J. Browne, "Comparing Narrowband and Wideband Channels," 07 February 2018. [Online]. Available: <https://www.mwrf.com/technologies/systems/article/21848973/comparing-narrowband-and-wideband-channels>. [Accessed 10 October 2020].
- [47] Federal Communications Commission, "Revision of Part 15 of the Commission's Rules Regarding Ultra-Wideband Transmission Systems," 2002.

- [48] Federal Communications Commission, "UNDERSTANDING THE FCC REGULATIONS FOR LOW-POWER, NON-LICENSED TRANSMITTERS," Office of Engineering and Technology, Columbia, MD, 1993.
- [49] M. Yavari and B. G. Nickerson, "Ultra Wideband Wireless Positioning Systems," 2014.
- [50] H. Chen, M. Chen, J. Zhang and S. Xie, "UWB monocycle and doublet pulses generation in optical domain," in *2007 International Topical Meeting on Microwave Photonics*, 2007.
- [51] I. Oppermann, M. Hämäläinen and J. Linatti, *UWB Theory and Applications*, John Wiley & Sons, Ltd, 2004.
- [52] P. Y. K. Cheung, "Lecture 3: Frequency Domain Analysis and Fourier Transform," Imperial College London - Dyson School of Design Engineering, 13 January 2020. [Online]. Available: [http://www.ee.ic.ac.uk/pcheung/teaching/DE2\\_EE/Lecture%203%20-%20Fourier%20Transform%20\(x1\).pdf](http://www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/Lecture%203%20-%20Fourier%20Transform%20(x1).pdf). [Accessed 10 October 2020].
- [53] F. Zafari, A. Gkelias and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," 2019.
- [54] M. Viot and M. Gross, "Enabling sub 10cm positioning accuracy," *Electronic Engineering Times Europe*, pp. 25-26, July/August 2014.
- [55] T. Matila, M. Kosamo, T. Patana, P. Jakkula, T. Hirvonen and I. Oppermann, "UWB Antennas," in *UWB Theory and Applications*, John Wiley & Sons, 2004, pp. 129-156.
- [56] X. L. Liang, *Ultra Wideband - Current Status and Future Trends*, M. A. Matin, Ed., IntechOpen, 2012.
- [57] D. Yee, "Ultra-Wideband Part II," 2015. [Online]. Available: <https://slideplayer.com/slide/5147619/>. [Accessed 13 April 2021].
- [58] N. Dahad, "Qorvo Acquires UWB Chip Provider Decawave for \$400m," 2020.
- [59] T. Peynot, J. Underwood and S. Scheduling, "Towards Reliable Perception for Unmanned Ground Vehicles in Challenging Conditions," in *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, 2009.
- [60] B. M.G. Cheetham, "Section 7: The Shannon-Hartley Theorem," 28 March 2006. [Online]. Available: [http://www.cs.man.ac.uk/~barry/mydocs/CS3282/Notes/DC06\\_7.pdf](http://www.cs.man.ac.uk/~barry/mydocs/CS3282/Notes/DC06_7.pdf). [Accessed 12 April 2021].
- [61] E. Blumenthal, "5G's many names explained: Don't fall for the carrier marketing fluff," Cnet, 26 February 2021. [Online]. Available: <https://www.cnet.com/how-to/5gs-many-names-explained-dont-fall-for-carrier-marketing-fluff/>. [Accessed 12 April 2021].
- [62] R. Triggs, "5G mmWave: Facts and fictions you should definitely know," Android Authority, 18 February 2019. [Online]. Available: <https://www.androidauthority.com/what-is-5g-mmwave-933631/>. [Accessed 13 April 2021].
- [63] C. Kopp, "Ultra Wide Band Modulation," September 2000. [Online]. Available: <http://www.airspacepower.net/AC-0900.html>. [Accessed 12 April 2021].
- [64] T. W. Barrett, "History of UltraWideBand (UWB) Radar & Communications: Pioneers and Innovators," in *Progress In Electromagnetics Symposium 2000 (PIERS2000)*, Cambridge, MA, 2000.

- [65] R. A. Scholtz, D. M. Pozar and W. Namgoong, "Ultra-Wideband Radio," *EURASIP Journal on Applied Signal Processing*, pp. 252-272, 2005.
- [66] L. Jaulin, "Instantaneous Localization," in *Mobile Robotics*, 2015, pp. 171-196.
- [67] Merriam-Webster, "Definition of trilateration".
- [68] G. Saud, "Global positioning system (gps)," Slideshare, 22 August 2015. [Online]. Available: <https://www.slideshare.net/gokulsaud/global-positioning-system-gps-51942739>. [Accessed 11 October 2020].
- [69] GIS Geography, "Trilateration vs Triangulation – How GPS Receivers Work," 24 February 2020. [Online]. Available: <https://gisgeography.com/trilateration-triangulation-gps/>. [Accessed 11 October 2020].
- [70] Bitcraze, "Loco Positioning system," [Online]. Available: <https://www.bitcraze.io/documentation/system/positioning/loco-positioning-system/>. [Accessed 09 April 2021].
- [71] J. Porto, "cyphyhouse/Decawave," 19 October 2020. [Online]. Available: <https://github.com/cyphyhouse/Decawave>. [Accessed 19 April 2021].
- [72] I. Dotlic, A. Connell, H. Ma, J. Clancy and M. McLaughlin, "Angle of Arrival Estimation Using Decawave DW1000 Integrated Circuits," in *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*, Bremen, Germany, 2017.
- [73] A. R. J. Ruiz and F. S. Granja, "Comparing Ubisense, BeSpoon, and DecaWave UWB Location Systems: Indoor Performance Analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 8, pp. 2106-2117, 03 April 2017.
- [74] B. Campbell, P. Dutta, B. Kempke, Y.-S. Kuo and P. Pannuto, "DecaWave: Exploring State of the Art Commercial Localization," Microsoft Indoor Localization Competition, Seattle, 2015.
- [75] F. Hammer, R. Yudianto, K. Neumann, M. Pichler, J. Cockx, C. Niestroj and F. Petré, "Performance Evaluation of 3D-Position Estimation Systems," *IEEE Sensors Journal*, vol. 16, no. 16, pp. 6416 - 6424, 15 August 2016.
- [76] J. Kulmer, S. Hinteregger, B. Großwindhager, M. Rath, M. S. Bakr, E. Leitinger and K. Witrisal, "Using DecaWave UWB transceivers for high-accuracy multipath-assisted indoor positioning," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, Paris, 2017.
- [77] Y. Chen, S. Zhao and J. A. Farrell, "Computationally Efficient Carrier Integer Ambiguity Resolution in MultiePOCH GPS/INS: A Common-Position-Shift Approach," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1541 - 1556, 08 December 2015.
- [78] T.-M. Nguyen, A. H. Zaini, K. Guo and L. Xie, "An Ultra-Wideband-based Multi-UAV Localization System in GPS-denied Environments," in *International Micro Air Vehicle Conference and Competition 2016 (IMAV 2016)*, 2016.
- [79] J. González, J.L. Blanco, C. Galindo, A. Ortiz-de-Galisteo, J.A. Fernández-Madrigal, F.A. Moreno and J.L. Martínez, "Mobile robot localization based on Ultra-Wide-Band ranging: A particle filter approach," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 496-507, 31 May 2009.

- [80] K. Guo, Z. Qiu, C. Miao, A. H. Zaini, C.-L. Chen, W. Meng and L. Xie, "Ultra-Wideband-Based Localization for Quadcopter Navigation," *Unmanned Systems*, vol. 04, no. 01, pp. 23-24, 2016.
- [81] Decawave, "Our Products," [Online]. Available: <https://www.decawave.com/products/>. [Accessed 10 April 2021].
- [82] Pozyx, "Kits | Our solutions that provide accurate positioning and motion information," [Online]. Available: <https://store.pozyx.io/>. [Accessed 10 April 2021].
- [83] J. Černohorský, P. Jandura and O. Mach, "Mobile robot localization and object description," in *2017 18th International Carpathian Control Conference (ICCC)*, Sinaia, Romania, 2017.
- [84] T. Conceição, F. N. d. Santos, P. Costa and A. P. Moreira, "Robot Localization System in a Hard Outdoor Environment," in *ROBOT 2017: Third Iberian Robotics Conference*, 2017.
- [85] B. McLoughlin, J. Cullen, A. Shaw and F. Bezomb, "Towards an Unmanned 3D Mapping System Using UWB Positioning," in *TAROS 2018: Towards Autonomous Robotic Systems*, 2018.
- [86] A. Jaikumar, "An Investigative Research on Available Localization Systems, Technologies, Applicability, Scalability. Setup & Testing of Wiser's kit, Data Extraction and Analysis.," 2018.
- [87] Decawave, "Partners," [Online]. Available: <https://www.decawave.com/partners/>. [Accessed 17 September 2018].
- [88] Decawave Ltd., "DW1000 Radio IC," [Online]. Available: <https://www.decawave.com/product/dw1000-radio-ic/>. [Accessed 10 April 2021].
- [89] Decawave Ltd., *DW1000IEEE802.15.4-2011 UWB Transceiver*, Dublin, 2016.
- [90] R. Wilson, "Location device work indoors with 10cm accuracy," *ElectronicsWeekly*, [Online]. Available: <https://www.electronicsworld.com/news/products/analog/location-device-work-indoors-with-10cm-accuracy-2013-11/>. [Accessed 10 April 2021].
- [91] Decawave (Qorvo), *DWM3000 IEEE 802.15.4-z UWB Transceiver Module*, 2020.
- [92] Decawave (Qorvo), "DW3000," [Online]. Available: <https://www.decawave.com/product/decawave-dw3000-ic/>. [Accessed 10 April 2021].
- [93] Decawave Ltd, "Frequently Asked Questions," [Online]. Available: <https://www.decawave.com/faq/>. [Accessed 10 April 2021].
- [94] Decawave Ltd, *APS014 APPLICATION NOTE: ANTENNA DELAY CALIBRATION OF DW1000-BASED PRODUCTS AND SYSTEMS*, 2018.
- [95] B. Tay, W. Liu, H. F. Chai and P. H. J. Chong, "Distributed RSSI-Sharing for Two-Way Ranging Base Station Selection," in *2010 11th International Conference on Control Automation Robotics & Vision*, Singapore, 2010.
- [96] Decawave Ltd, *APS016 APPLICATION NOTE: MOVING FROM TREK1000 TO A PRODUCT*, 2018.
- [97] G.-M. Hoang, *Localization via Ultra Wide Band Radio for Wireless Body Area Sensor Networks*, M. Gautier and A. Courty, Eds., Rennes: Université de Rennes, 2014.

- [98] J. Li, Y. Bi, K. Li, K. Wang, F. Lin and B. M. Chen, "Accurate 3D Localization for MAV Swarms by UWB and IMU Fusion," 23 November 2018. [Online]. Available: <https://github.com/lijx10/uwb-localization>. [Accessed 27 April 2021].
- [99] E. Erdogan, "indoor\_localization," 08 November 2019. [Online]. Available: [http://wiki.ros.org/indoor\\_localization](http://wiki.ros.org/indoor_localization). [Accessed 27 April 2021].
- [100] K. Shadi, "Localization 0.1.7," 23 December 2018. [Online]. Available: <https://pypi.org/project/Localization/>. [Accessed 27 April 2021].
- [101] A. Wallar, braraki and D. Chong, "DecaWave Localization ROS Node," [Online]. Available: [https://github.com/mit-drl/decawave\\_localization/](https://github.com/mit-drl/decawave_localization/). [Accessed 27 April 2021].
- [102] S. G. Johnson, "The NLOpt nonlinear-optimization package," [Online]. Available: <https://nlopt.readthedocs.io/en/latest/>. [Accessed 27 April 2021].
- [103] Decawave Ltd., "TREK1000: Two-Way-Ranging (TWR) RTLS IC Evaluation Kit," 2015.
- [104] Open Robotics, "About ROS," [Online]. Available: <https://www.ros.org/about-ros/>. [Accessed 19 April 2021].
- [105] *How to Start a Robot Revolution / Part 1 / Breaking the Wheel*. [Film]. Red Hat, 2020.
- [106] Open Robotics, "ros\_comm," 11 February 2015. [Online]. Available: [http://wiki.ros.org/ros\\_comm?distro=noetic](http://wiki.ros.org/ros_comm?distro=noetic). [Accessed 20 April 2021].
- [107] Open Robotics, "ROS geometry\_msgs," [Online]. Available: [http://wiki.ros.org/geometry\\_msgs](http://wiki.ros.org/geometry_msgs). [Accessed 19 April 2021].
- [108] Vicon Motion Systems Ltd, "WHAT IS MOTION CAPTURE," [Online]. Available: <https://www.vicon.com/about-us/what-is-motion-capture/>. [Accessed 19 April 2021].
- [109] Vicon Motion Systems Ltd, "TRACKER," [Online]. Available: <https://www.vicon.com/software/tracker/>. [Accessed 19 April 2021].
- [110] P. Bovbel, "vrpn\_client\_ros," Open Robotics, 28 June 2017. [Online]. Available: [http://wiki.ros.org/vrpn\\_client\\_ros](http://wiki.ros.org/vrpn_client_ros). [Accessed 19 April 2021].
- [111] P. Krzyzanowski, "Clock Synchronization," 2016.
- [112] C. S. V. Gutiérrez, L. U. S. Juan, I. Z. Ugarte, I. M. Goenaga, L. A. Kirschgens and V. M. Vilches, "Time Synchronization in modular collaborative robots," 2018.
- [113] Vicon Motion Systems Ltd, "Vicon DataStream SDK 1.11.0 Developer's Guide," 2020.
- [114] Microsoft, "How the Windows Time Service Works," 08 May 2018. [Online]. Available: <https://docs.microsoft.com/en-us/windows-server/networking/windows-time-service/how-the-windows-time-service-works>. [Accessed 19 April 2021].
- [115] Microsoft, "Configuring Systems for High Accuracy," 08 May 2018. [Online]. Available: <https://docs.microsoft.com/en-us/windows-server/networking/windows-time-service/configuring-systems-for-high-accuracy>. [Accessed 19 April 2021].
- [116] Canonical, "Time Synchronization," [Online]. Available: <https://ubuntu.com/server/docs/network-ntp>. [Accessed 19 April 2021].
- [117] J. Faust, V. Pradeep and D. Thomas, "message\_filters," Open Robotics, 14 August 2018. [Online]. Available: [http://wiki.ros.org/message\\_filters#ApproximateTime\\_Policy](http://wiki.ros.org/message_filters#ApproximateTime_Policy). [Accessed 19 April 2021].



- [118] J. Faust, "message\_filters::sync::ApproximateTime," Open Robotics, 28 July 2010. [Online]. Available: [http://wiki.ros.org/message\\_filters/ApproximateTime](http://wiki.ros.org/message_filters/ApproximateTime). [Accessed 19 April 2021].
- [119] A. Gramfort, "Matlab mesh to PDF with 3D interactive object," 10 January 2010. [Online]. Available: <https://www.mathworks.com/matlabcentral/fileexchange/25383-matlab-mesh-to-pdf-with-3d-interactive-object>. [Accessed 11 May 2020].
- [120] I. Filippidis, "Export figure to 3D interactive PDF," 27 February 2015. [Online]. Available: <https://www.github.com/johnyf/fig2u3d>. [Accessed 11 May 2020].
- [121] Clearpath Robotics, "Jackal," [Online]. Available: <https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/>. [Accessed 20 April 2021].
- [122] Trimble, "Trimble® S3 Total Station: User Guide," Trimble Navigation Limited, Sunnyvale, CA, 2010.
- [123] G. Mishra, "Total Station in Surveying: Operation, Uses and Advantages," The Constructor, [Online]. Available: <https://theconstructor.org/surveying/total-station-operation-uses-advantage/6605/>. [Accessed 20 April 2020].
- [124] J. Mahun, "Chapter B. Vertical and Horizontal," 20 July 2019. [Online]. Available: <https://jerrymahun.com/index.php/home/open-access/i-basic-principles/81-i-b>. [Accessed 20 April 2021].
- [125] M. P. Tully Foote, "ROS REP 103: Standard Units of Measure and Coordinate Conventions," 07 October 2010. [Online]. Available: <https://www.ros.org/reps/rep-0103.html>. [Accessed 20 April 2021].
- [126] J. Mahun, "Chapter F. Vertical Angles," 11 October 2018. [Online]. Available: <https://jerrymahun.com/index.php/home/open-access/i-basic-principles/85-i-f-vert-angles>. [Accessed 20 April 2021].
- [127] J. Mahun, "Chapter G. Horizontal Angles," 11 October 2018. [Online]. Available: <https://jerrymahun.com/index.php/home/open-access/i-basic-principles/88-i-g-horizontal-angles>. [Accessed 20 April 2021].
- [128] Argo, "J8 XTR," Ontario Drive & Gear Ltd., 2017. [Online]. Available: <https://www.argo-xtr.com/index.php/xtr-robots/j8-atlas-xtr/>. [Accessed 21 April 2021].
- [129] M. Jacobson, "Absolute Orientation - Horn's method," Xoran Technologies, Inc., 02 September 2015. [Online]. Available: <https://www.mathworks.com/matlabcentral/fileexchange/26186-absolute-orientation-horn-s-method>. [Accessed 21 April 2021].
- [130] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America*, vol. 4, no. 4, pp. 629-642, 25 November 1986.
- [131] Trimble, Inc., "Trimble SX10," [Online]. Available: <https://geospatial.trimble.com/products-and-solutions/trimble-sx10>. [Accessed 22 April 2021].
- [132] Trimble, Inc., "Trimble SX10 Datasheet," 2017.
- [133] KVH Industries, Inc., "GEO-FOG 3D," [Online]. Available: <https://www.kvh.com/admin/products/gyros-imus-inss/ins/geo-fog-3d/commercial-geo-fog-3d>. [Accessed 24 April 2021].

- [134] The MathWorks, Inc., "MATLAB: sph2cart()," [Online]. Available: <https://www.mathworks.com/help/matlab/ref/sph2cart.html>. [Accessed 24 April 2021].
- [135] Decawave, Ltd., "Beta PDoA kit User Manual," 2018.
- [136] Tera Term Project, "Tera Term Home Page," [Online]. Available: <https://ttssh2.osdn.jp/index.html.en>. [Accessed 25 April 2021].
- [137] The MathWorks, Inc., "ROS Custom Message Support," [Online]. Available: <https://www.mathworks.com/help/ros/ug/ros-custom-message-support.html>. [Accessed 25 April 2021].
- [138] The MathWorks, Inc., "Polynomial Models," [Online]. Available: <https://www.mathworks.com/help/curvefit/polynomial.html#bryq46g>. [Accessed 26 April 2021].
- [139] P. M. Maxim, S. Hettiarachchi, W. M. Spears, D. F. Spears, J. Hamann, T. Kunkel and C. Speiser, "Trilateration Localization for Multi-Robot Teams," in *Proceedings of the Fifth International Conference on Informatics in Control, Automation and Robotics, Robotics and Automation*, 2008.
- [140] P. Cotera, M. Velazquez, D. Cruz, L. Medina and M. Bandala, "Indoor Robot Positioning using an Enhanced Trilateration Algorithm," *International Journal of Advanced Robotic Systems*, vol. 13, no. 110, 20 March 2016.
- [141] Garmin Ltd., "Understanding the Accuracy of the GPS Elevation Reading," [Online]. Available: <https://support.garmin.com/en-US/?faq=QPc5x3ZFUv1QyoxITW2vZ6>. [Accessed 26 April 2021].
- [142] Federal Communications Commission, "Revision of Part 15 of the Commission's Rules Regarding Ultra-Wideband Transmission Systems," 2002.
- [143] C. Connell, "What's The Difference Between Measuring Location By UWB, Wi-Fi, and Bluetooth?," 6 February 2015. [Online]. Available: <https://www.electronicdesign.com/technologies/communications/article/21800581/whats-the-difference-between-measuring-location-by-uwf-wifi-and-bluetooth>. [Accessed 15 November 2020].
- [144] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. A. Al-Ammar and H. S. Al-Khalifa, "Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances," *Sensors*, vol. 16, no. 707, 16 May 2016.
- [145] Ultra-wideband by INFISOFT, "Application Examples," [Online]. Available: <https://www.ultrawideband.io/en/examples.php>.
- [146] Locatify, "Ultrawideband," 2017. [Online]. Available: <https://locatify.com/blog/tag/ultrawideband/>. [Accessed 15 November 2020].
- [147] N. Agarwal, J. Basch, P. Beckmann, Bharti and e. al., "Algorithms for GPS operation indoors and downtown," *GPS Solutions*, vol. 6, pp. 149-160, 2002.
- [148] Caterpillar, "ACCUGRADE GPS," [Online]. Available: [https://www.cat.com/en\\_US/support/operations/technology/earth-moving-solutions/accugrade-grade-control-system/gps.html](https://www.cat.com/en_US/support/operations/technology/earth-moving-solutions/accugrade-grade-control-system/gps.html). [Accessed 15 Nov 2020].

- [149] Trimble, "GNSS Systems," [Online]. Available: <https://geospatial.trimble.com/products-and-solutions/gnss-systems>. [Accessed 15 Nov 2020].
- [150] M. B., "Led by Samsung and Apple: Is UWB The Next Big Smartphone Tech?," 14 Oct 2020. [Online]. Available: <https://www.sciencetimes.com/articles/27710/20201014/led-samsung-apple-uw-next-big-smartphone-tech.htm>. [Accessed 15 Nov 2020].
- [151] L. Mearian, "Ultra Wideband (UWB) explained (and why it's in the iPhone 11)," 31 Dec 2019. [Online]. Available: <https://www.computerworld.com/article/3490037/ultra-wideband-explained-and-why-its-in-the-iphone-11.html>. [Accessed 15 Nov 2020].
- [152] Qorvo, "Qorvo® Completes Acquisition of Decawave," 24 Feb 2020. [Online]. Available: <https://www.qorvo.com/newsroom/news/2020/qorvo-completes-acquisition-of-decawave>. [Accessed 15 Nov 2020].
- [153] W. Gallagher, "Apple supplier Qorvo acquires ultra-wideband chipmaker Decawave," Feb 2020. [Online]. Available: <https://appleinsider.com/articles/20/01/30/apple-supplier-qorvo-acquires-ultra-wideband-chipmaker-decawave>. [Accessed 15 Nov 2020].
- [154] Apple Inc., "Apple introduces AirTag," 20 April 2021. [Online]. Available: <https://www.apple.com/newsroom/2021/04/apple-introduces-airtag/>. [Accessed 26 April 2021].
- [155] Humatics, "Transportation," [Online]. Available: <https://humatics.com/industries/transportation/>. [Accessed 15 Nov 2020].
- [156] D. Edwards, "Construction and mining 'next frontier' for robotics adoption, says research company," 22 Jan 2019. [Online]. Available: <https://roboticsandautomationnews.com/2019/01/22/construction-and-mining-next-frontier-for-robotics-adoption-says-research-company/20655/>. [Accessed 15 Nov 2020].
- [157] L.Nomdedeu, J.Sales, R.Marín, E.Cervera and J.S.Aez, "Sensing capabilities for mobile robotics," in *Using Robots in Hazardous Environments: Landmine Detection, De-Mining and Other Applications*, Woodhead Publishing, 2011, pp. 125-146.

## APPENDIX A: GitHub Repository

All relevant materials created by the author to realize the results of this thesis are contained in a public GitHub repository: <https://github.com/nicholasAWwright/UWB-Thesis-MSME-UIUC>